natural

Natural for DB2	Version 4.1.2	Natural for DB2

software AG

This document applies to Natural for DB2 Version 4.1.2 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Table of Contents

Natural for DB2 - Overview

With the Natural interface to DB2, a Natural user can access data in a DB2 database. Natural for DB2 is supported under Com-plete, CICS, IMS/TM, batch mode, and TSO.

In general, there is no difference between using Natural with DB2 and using it with Adabas, VSAM or DL/I. The Natural interface to DB2 allows Natural programs to access DB2 data, using the same Natural DML statements that are available for Adabas, VSAM, and DL/I. Therefore, programs written for DB2 tables can also be used to access Adabas, VSAM, or DL/I databases. In addition, Natural SQL statements are available.

All operations requiring interaction with DB2 are performed by the Natural interface module.

Natural Version 4.1 or above is required for Version 4.1 of Natural for DB2.

Note:
The term "file server" used in this documentation only refers to the Natural file server for DB2.

Related Documentation
For information on logging SQL statements contained in a Natural program, refer to DBLOG Utility in the Natural Utilities documentation.

This documentation covers:

General Information	Information on how to access DB2 tables, on the integration with Software AG's Data Dictionary Predict, and on error messages related to DB2
Installing Natural for DB2	Installation of the Natural interface to DB2 and description of the Natural for DB2 parameter module.
Natural Tools for DB2	Various utilities and commands to: • maintain application plans and packages • maintain the DB2 catalog • define and list stored procedures • generate and issue interactive SQL statements • retrieve data from system tables • display and modify environment settings • list and explain plan tables • display file server statistics • issuing DB2 commands from Natural • display and explain generated SQL statements • display Natural programs that access DB2
DDM Generation	Generation of Natural data definition modules (DDMs) using the SQL Services function of the Natural SYSDDM utility.
Dynamic and Static SQL Support	Internal handling of dynamic statements, creation and execution of static DBRMs, mixed dynamic/static mode, and application plan switching in the various supported environments.
Statements and System Variables	Special considerations on Natural DML statements, Natural SQL statements and Natural system variables with DB2. In addition, the Natural for DB2 enhanced error handling is discussed.
Natural Stored Procedures and UDFs	Processing Natural stored procedures and Natural user-defined functions (UDFs).
Interface Subprograms	Several Natural and non-Natural subprograms to be used for various purposes.
Natural File Server for DB2	Description of the Natural File Server in the various supported environments.
Environment-Specific Considerations	Special considerations on the various environments supported by Natural for DB2.

NDB - General Information

This section covers the following topics:

- Accessing a DB2 Table
- Integration with Predict
- Natural System Messages Related to DB2

Accessing a DB2 Table

To be able to access a DB2 table with a Natural program

1. Use the Natural Tools for DB2 to define a DB2 table.
2. Use Predict or the SQL Services function of the Natural SYSDDM utility to create a Natural DDM of the defined DB2 table.
3. Once you have defined a DDM for a DB2 table, you can access the data stored in this table by using a Natural program.

The Natural interface to DB2 translates the statements of a Natural program into SQL statements.

Natural automatically provides for the preparation and execution of each statement. In dynamic mode, a statement is only prepared once (if possible) and can then be executed several times. For this purpose, Natural internally maintains a table of all prepared statements.

Almost the full range of possibilities offered by the Natural programming language can be used for the development of Natural applications which access DB2 tables. For a number of Natural DML statements, however, there are certain restrictions and differences as far as their use with DB2 is concerned; see Natural DML Statements as described in Statements and System Variables. In the Natural Statements documentation, you can find notes on Natural usage with DB2 attached to the descriptions of the statements concerned.

As there is no DB2 equivalent to Adabas ISNs (Internal Sequence Numbers), any Natural features which use ISNs are not available when accessing DB2 tables with Natural.

For SQL-eligible databases, in addition to the Natural DML statements, Natural provides SQL statements as described in Statements and System Variables. In the Natural Statements documentation you can find a detailed description of these statements.

Integration with Predict

Since Predict supports DB2, direct access to the DB2 catalog is possible via Predict and information from the DB2 catalog can be transferred to the Predict dictionary to be integrated with data definitions for other environments.

DB2 databases, tables and views can be incorporated and compared, new DB2 tables and views can be generated, and Natural DDMs can be generated and compared. All DB2-specific data types and the referential integrity of DB2 are supported. See the relevant Predict documentation for details.

In addition, the Predict active references support static SQL for DB2.

Natural System Messages Related to DB2

The message number ranges of Natural system messages related to DB2 are 3275 - 3286, 3700-3749, and 7386-7395.

Installing Natural for DB2 - Overview

This section describes how to install the Natural interface to DB2 (in the remainder of this section also referred to as NDB) in the various environments supported.

The installation procedures contain a number of options that depend on the TP monitor being used as well as on other site requirements.

- General Information
- Installation Tape
- Installation Procedure
- Installation Verification
- Natural Parameter Modification for DB2
- Parameter Module NDBPARM
- Special Requirements for Natural Tools for DB2
- Natural for DB2 Server Stub

Installing NDB - General Information

This section covers the following topics:

- Installation Jobs
- Using System Maintenance Aid
- Prerequisites

Installation Jobs

The installation of Software AG products is performed by installation jobs. These jobs are either created manually or generated by System Maintenance Aid (SMA).

For each step of the installation procedure described later in the section Installing Natural for DB2, the job number of a job performing the respective task is indicated. This job number refers to an installation job generated by SMA. If you are not using SMA, an example job of the same number is provided in the job library on the NDB installation tape; you must adapt this example job to your requirements. Note that the job numbers on the tape are preceded by a product code (for example, NDBI070).

Using System Maintenance Aid

For information on the use of Software AG's System Maintenance Aid for the installation process, refer to the System Maintenance Aid documentation.

Prerequisites

- Base Natural Version 4.1 or above must be installed first; you cannot install Natural 4.1 and Natural for DB2 Version 4.1 at the same time.
- The Software AG Editor must be installed (see Installing the Software AG Editor in the Natural Installation Guide for Mainframes).
- If you want to use the DB2 DL/I batch support (DSNMTV01), the Natural interface to DL/I is required.

Further product/version dependencies are specified under Natural and Other Software AG Products and Operating/Teleprocessing Systems Required in the current Natural Release Notes for Mainframes.

NDB - Installation Tape

The installation tape contains the datasets listed in the table below. The sequence of the datasets is shown in the Report of Tape Creation which accompanies the installation tape.

Dataset Name	Contents
NDB*nnn*.SRCE	NDB source modules
NDB*nnn*.LOAD	NDB load modules
NDB*nnn*.INPL	NDB utility programs in INPL format
NDB*nnn*.ERRN	NDB error messages
NDB*nnn*.JOBS	NDB installation jobs

The notation *nnn* in dataset names represents the version number of the product.

Copying the Tape Contents to Disk

If you are using System Maintenance Aid (SMA), refer to the SMA documentation (included on the current edition of the Natural documentation CD).

If you are **not** using SMA, follow the instructions below.

This section explains how to:

- Copy data set COPY.JOB from tape to disk.
- Modify this data set to conform with your local naming conventions.

The JCL in this data set is then used to copy all data sets from tape to disk.

If the datasets for more than one product are delivered on the tape, the dataset COPY.JOB contains the JCL to unload the datasets for all delivered products from the tape to your disk.

After that, you will have to perform the individual install procedure for each component.

Step 1 - Copy data set COPY.JOB from tape to disk

The data set COPY.JOB (label 2) contains the JCL to unload all other existing data sets from tape to disk. To unload COPY.JOB, use the following sample JCL:


```
//SAGTAPE JOB SAG,CLASS=1,MSGCLASS=X
//* --------------------------------
//COPY EXEC PGM=IEBGENER
//SYSUT1 DD DSN=COPY.JOB,
// DISP=(OLD,PASS),
// UNIT=(CASS,,DEFER),
// VOL=(,RETAIN,SER=<Tnnnnn>),
// LABEL=(2,SL)
//SYSUT2 DD DSN=<hilev>.COPY.JOB,
// DISP=(NEW,CATLG,DELETE),
// UNIT=3390,VOL=SER=<vvvvvv>,
// SPACE=(TRK,(1,1),RLSE),
// DCB=*.SYSUT1
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//
```

Where:

<*hilev*>	is a valid high level qualifier
<*Tnnnnn*>	is the tape number
<*vvvvvv*>	is the desired volser

Step 2 - Modify COPYTAPE.JOB

Modify the COPYTAPE.JOB to conform with your local naming conventions and set the disk space parameters before submitting this job:

- Set HILEV to a valid high level qualifier.
- Set LOCATION to a storage location.
- Set EXPDT to a valid expiration date.

Step 3 - Submit COPY.JOB

Submit COPY.JOB to unload all other data sets from the tape to your disk.

NDB - Installation Procedure - Overview

This section describes how to install Natural for DB2 in various environments:

- Steps Common to all Environments
- Steps Specific to CICS
- Steps Specific to Com-plete
- Steps Specific to IMS/TM
- Steps Specific to TSO

NDB - Steps Common to all Environments

The following steps describe the procedure for installing the components of NDB that are common to all environments.

Step 1: Allocate DBRM library for use with NDB

Allocate a PDS as DBRM (database request module) library. The size of this dataset and the number of directory entries depend on the particular site (5 tracks and 20 directory blocks must be adequate for most environments). The PDS must have a fixed-block record format and a record length of 80.

Any standard dataset name can be used for this DBRM library; however, this installation procedure assumes that the name SAGLIB.SMADBRM is used.

Step 2: Generate NDB I/O module NDBIOMO - Job I055, Step 1600

By executing a standard Natural batch job, this step generates the assembly source for NDBIOMO from the member NDBIOTM.

This batch job invokes the Natural program NDBGENI, which is loaded with INPL during the base Natural installation. NDBGENI contains the following two parameters, which you can modify to meet your specific requirements:

- the DB-environment parameter, which must be set to:
  - DB2V5 if you are running DB2 Version 5
  - DB2V6 if you are running DB2 Version 6
  - DB2V7 if you are running DB2 Version 7
- the parameter to specify the number of statements for dynamic access.

NDBIOMO performs the dynamic access to DB2 and contains all necessary EXEC SQL statements. In addition, it contains some special SQL statements which cannot be executed in dynamic mode.

An output report is created by this job. Check the report for successful job completion. In addition, a condition code of 0 indicates normal completion.

Step 3: Assemble and link NDBIOMO - Job I055, Step 1610

Precompile, assemble and link NDBIOMO.

Note:
The link-edit step receives a condition code of 4 because of unresolved references for DSNHLI. This is normal and can be ignored.

Step 4: Create NDB plan - Job I055, Step 1630

If desired, change library names and plan name to meet site requirements.

Step 5: Modify, assemble and link NDB parameter module Job I055, Steps 1640/1650 or 1660/1670 or 1675/1676

The NDB parameter module contains the macro NDBPRM with parameters specific to the Natural interface to DB2.

You can generally use the default values for all parameters. Modify only the values of the parameters whose default values do not suit your requirements.

The individual parameters are described in the section Parameter Module NDBPARM.

- When the file server is **not** to be used:
  Execute the Steps 1640 and 1650; the resulting parameter module is called NDBPARM.
- When the file server is to be used:
  Execute the Steps 1660 and 1670; the resulting parameter module is called NDBPARMF.
- When the file server uses the Software AG Editor buffer pool as storage medium:
  Execute the Steps 1675 and 1676; the resulting parameter module is called NDBPARME.

Step 6: Link-edit NATGWDB2 - Job I055, Step 1680

Link-edit the environment-independent NDB nucleus NATGWDB2. Verify that the INCLUDE cards refer to the corresponding DD names for the load libraries.

Step 7: Modify, assemble and link NATPARM

Adapt your Natural parameter module NATPARM by adding parameters specific to Natural for DB2 (see Natural Parameter Modification for DB2) and reassemble NATPARM.

Step 8: Relink your Natural nucleus

Natural for DB2 basically consists of:

- An environment-independent nucleus, which can be shared by multiple environments and which is therefore LPA-eligible.
- Environment-dependent components, which must be linked to the appropriate Natural environment-dependent interface.

Modify the JCL used to link your Natural shared nucleus by adding the following INCLUDE card:

INCLUDE SMALIB(NATGWDB2)	Environment-independent NDB nucleus from Step 6

Modify the JCL used to link your Natural environment-dependent nucleus by adding the following INCLUDE cards and the corresponding DD statements:

INCLUDE SMALIB(NDBPARM)	NDB parameter module created in Step 5
INCLUDE SMALIB(NDBIOMO)	NDB I/O module created in Step 3
INCLUDE DSNLIB(DSNTIAR)	SQL Error Message Module
INCLUDE *xxxxxxxx*(*yyyyyyyy*)	Environment-dependent DB2 Interface (see below)

If you want to use the Natural File Server, include SMALIB(NDBPARMF) or SMALIB(NDBPARME) instead of SMALIB(NDBPARM); see also Step 5 above.

Depending on your environment(s), INCLUDE the appropriate environment-specific language interface *yyyyyyyy* in the library *xxxxxxxx* as shown in the following table:

Interface	Library	Environment
DSNALI	DSNLIB	Under TSO and in batch mode without running under the control of the DSN command processor (that is, with CAF).
DSNRLI	DSNLIB	WLM (Workload Manager) stored procedure address space.
DSNELI	DSNLIB	Under TSO and in batch mode when running under the control of the DSN command processor.
DSNCLI	DSNLIB	Under CICS
DFSLI000	IMSLIB	Under IMS/TM (MPP and BMP) and in batch mode by using the DB2 DL/I batch support (DSNMTV01).
NDBCOM	NDBLIB	Under Com-plete.

Note:
If you want to use NDB in various environments (that is, with different TP monitors), you must repeat this step for each of these environments.

Instead of link-editing your Natural nucleus in the way described above, you have the following alternatives:

1. If you do not use a Natural shared nucleus, all modules must be included in the link-edit of the Natural nucleus.
2. Remove NATGWDB2 from the link-edit of the Natural shared nucleus and run it as a separate module with the mandatory entry name NATGWDB2. You can modify the name of the module created in Step 6. However, if you use a name different from NATGWDB2, this name must be specified as an alias name in an NTALIAS macro entry of the Natural parameter module. This way of link-editing only applies if the Natural Resolve CSTATIC Addresses feature (RCA) is used.
3. Include all modules in the link-edit job of a separate Natural parameter module with the mandatory entry name CMPRMTB. The name of the resulting module is arbitrary. This way of link-editing only applies if an alternative parameter module (PARM profile parameter) is used.
   If link-editing is done in this way, you can install NDB without having to modify your Natural nucleus or driver.

If link-editing is done according to number [2] or [3], the following applies:

TP Monitor	Requirement
CICS	The resulting module must be defined via a PPT entry or RDO. PPT entry: `DFHPPT TYPE=ENTRY,PROGRAM=`*`module-name`*`,PGMLANG=ASSEMBLER`
Com-plete	The resulting module must be defined as RESIDENTPAGE or reside in the LPA/(E)LPA.

Step 9: Load Natural objects into system file - Job I061, Step 1600

Before executing this step, change the CMWKF01 DD statement to point to the NDB*nnn*.INPL dataset.

In this step, the NDB system programs, maps and DDMs are loaded into the Natural system files. The INPL job loads objects into the Natural system libraries SYSDDM, SYSTEM and SYSDB2.

The NDB system programs **must** be loaded into the Natural 4.1. FNAT system file.

Warning:
Ensure that your newly created SYSDB2 library contains all necessary Predict interface programs, which are loaded into SYSDB2 when installing Predict (see the relevant Predict documentation).

Step 10: Load Natural error messages into system file - Job I061, Step 1620

Before executing this step, change the CMWKF02 DD statement to point to the NDB*nnn*.ERRN dataset.

This step executes a batch Natural job that runs an error load program by using the NDB*nnn*.ERRN dataset as input. The ERRLODUS job loads error messages into the library SYSERR on the FNAT system file.

The NDB error messages **must** be loaded into the Natural 4.1. FNAT system file.

Step 11: Create NDB server stub - Job I070, Steps 1604,1606,1608,1610

Create server stubs to execute Natural stored procedures and Natural user-defined functions. Natural for DB2 (NDB) server stubs are interface modules between the DB2 database system and the Natural server. In order to execute Natural stored procedures and Natural user-defined functions, the server stub needs to be installed.

There are two types of server stub:

1. The NDB server stub (module NDB41SRV, Steps 1604 and 1606).
   The server stub is used to execute Natural stored procedures and Natural user-defined functions.

   The IBM LE (Language Environment) runtime modules required must be linked to the NDB server stub module. Use the CALL option of the linkage editor and assign the LE runtime library as SYSLIB.
2. The NDB start server stub (module NDB41STR, Steps 1608 and 1610)
   The start server stub is used to start the Natural server environment(s) explicitly.

   The IBM LE (Language Environment) runtime modules required must be linked to the NDB start server stub module. Use the CALL option of the linkage editor and assign the LE runtime library as SYSLIB. Additionally, include the modules NDBSTRP (delivered with NDB) and NATCONFG (delivered with Natural) from NDB*nnn*.LOAD and NAT*nnn*.LOAD.

The NDB server stubs are generated from the NDBSTUB macro. You can generally use the default values for all parameters. Modify only the values of the parameters whose default values do not suit your requirements. The individual parameters are described in the section NDB Server Stub.

The resulting load modules have to be placed into a steplib library of the JCL used to execute the DB2 stored procedure address space.

For DB2 for OS/390 Version 5 and below, each Natural stored procedure or Natural user-defined function must be defined in the SYSIBM.SYSPROCEDURES table of the DB2 catalog, and the LOADMOD column must contain the name of the generated NDB server stub module NDB41SRV.

For DB2 UDB for OS/390 Version 6 and above, each Natural stored procedure or Natural user-defined function must be defined by a DB2 CREATE PROCEDURE or DB2 CREATE FUNCTION statement, where the name of the NDB server stub module NDB41SRV generated is specified as EXTERNAL NAME.

Step 12: Bind DBRM ROUTINEN into package - Job I070, Step 1615

Bind DBRM ROUTINEN into a package. DBRM ROUTINEN is contained in the collection SAGNDBROUTINENPACK and delivered with NDB. NDB needs this collection for accessing the DB2 catalog and retrieving the parameter descriptions of Natural stored procedures and Natural user-defined functions.

NDB - Steps Specific to CICS

This section describes how to install Natural for DB2 in a CICS environment. Ensure that your Natural/CICS thread size is large enough to contain the DB2SIZE; if you use the Natural Tools for DB2, an additional storage of 8 KB is required.

This section covers the following topics:

- Using Plan Selection by CICS RCT Entry Threads
- Using Plan Selection by Dynamic Plan Exit
- Using the File Server with VSAM

Using Plan Selection by CICS RCT Entry Threads

If you want fixed assignment of your transaction code to the DB2 plan, add an additional entry to your RCT, or create a DB2ENTRY with RDO if you have CICS TS installed.

Below is information on:

- Step 1: Modify, assemble and link CICS RCT or create DB2ENTRY

Step 1: Modify, assemble and link CICS RCT or create DB2ENTRY

Modify your RCT as follows (for any other parameters, refer to the relevant DB2 literature by IBM):

`DSNRCT TYPE=ENTRY,PLAN=`*`plan-name`*`,TXID=(`*`transaction-ID`*`)`

Or:

Define a DB2ENTRY with RDO (for parameters, refer to the relevant CICS literature by IBM:

```
DEFINE DB2ENTRY
OVERTYPE TO MODIFY                                      CICS RELEASE = 0530
CEDA  DEFine DB2Entry(          )
 DB2Entry       : DB2ENTR
 Group          : NCI
 DEscription    :
THREAD SELECTION ATTRIBUTES
 TRansid        : transaction-id
THREAD OPERATION ATTRIBUTES
 ACcountrec     : None              None ! TXid ! TAsk ! Uow
 AUTHId         :
 AUTHType       : Userid            Userid ! Opid ! Group ! Sign ! TErm
                                    ! TX
 DRollback      : Yes               Yes ! No
 PLAN           : plan-name
 PLANExitname   :
 PRIority       : High             High ! Equal ! Low
 PROtectnum     : 0005              0-2000
 THREADLimit    : 0005              0-2000
 THREADWait     : Pool              Pool ! Yes ! No
```

The *plan-name* must be the same as the name used in Step 4 - Create NDB plan (Installation Procedure, Steps Common to all Environments).

Using Plan Selection by Dynamic Plan Exit

If you want to perform plan selection by using the dynamic plan exit, perform the following steps:

- Step 1: Assemble CICS dynamic plan selection exit module NDBUEXT - Job I070, Step 1630
- Step 2: Link-edit CICS dynamic plan selection exit module NDBUEXT - Job I075, Step 1640
- Step 3: Modify, assemble and link CICS RCT or create DB2ENTRY

Step 1: Assemble CICS dynamic plan selection exit module NDBUEXT - Job I070, Step 1630

The sample exit NDBUEXT can be modified to use a default plan name if none has been specified prior to the first SQL call. Review the source code in the module NDBUEXT for details about specifying a default plan name.

Optionally modify the source module NDBUEXT.

Precompile, assemble and link NDBUEXT for CICS.

Note:
This step receives a condition code of 4 because of an unresolved external reference for DFHEAI0 and DFHEI1. This is normal and can be ignored.

Step 2: Link-edit CICS dynamic plan selection exit module NDBUEXT - Job I075, Step 1640

The resulting module NDBUEXT must be linked to the CICS load library and defined via a corresponding PPT entry or RDO.

PPT entry:

```
DFHPPT TYPE=ENTRY,PROGRAM=NDBUEXT,PGMLANG=ASSEMBLER
```

Step 3: Modify, assemble and link CICS RCT or create DB2ENTRY

Modify your RCT as follows (for any other parameters, refer to the relevant DB2 literature by IBM):

```
DSNRCT TYPE=POOL,PLNPGME=NDBUEXT,PLNEXIT=YES
```

The parameter PLNPGME must specify the same program as the NAME statement of the link step above.

Or:

Define a DB2ENTRY with RDO if you have CICS TS installed (for parameters, refer to the relevant CICS literature by IBM):

```
DEFINE DB2ENTRY
OVERTYPE TO MODIFY                                      CICS RELEASE = 0530
CEDA  DEFine DB2Entry(          )
 DB2Entry       : DB2ENTR
 Group          : NCI
 DEscription    :
THREAD SELECTION ATTRIBUTES
 TRansid        : transaction-id
THREAD OPERATION ATTRIBUTES
 ACcountrec     : None              None ! TXid ! TAsk ! Uow
 AUTHId         :
```



```
AUTHType       : Userid             Userid ! Opid ! Group ! Sign ! TErm
                                    ! TX
DRollback      : Yes                Yes ! No
PLAN           :
PLANExitname   : NDBUEXT
PRIority       : High              High ! Equal ! Low
PROtectnum     : 0005               0-2000
THREADLimit    : 0005               0-2000
THREADWait     : Pool               Pool ! Yes ! No
```

The parameter PLANExitname must specify the same program as the NAME statement of the link step above.

Alternatively or additionally, you can specify the plan exit program NDBUEXT with the PLANExitname parameter of POOL THREAD ATTRIBUTES of the DB2Conn resource definition of CICS TS.

Using the File Server with VSAM

If you want to use the Natural File Server (VSAM), perform the following additional steps:

- Step 1: Define VSAM dataset for file server - Job I008, Step 1610
- Step 2: Format file server dataset - Job I075, Step 1610
- Step 3: Modify, assemble and link CICS tables
- Step 4: Restart CICS

Step 1: Define VSAM dataset for file server - Job I008, Step 1610

Specify the size and the name of the VSAM RRDS that is to be used as the file server (see also Installing the File Server in Natural File Server for DB2).

Step 2: Format file server dataset - Job I075, Step 1610

Specify the five input parameters required to format the file server dataset (see also Installing the File Server in Natural File Server for DB2).

Step 3: Modify, assemble and link CICS tables

Shown below are sample additional CICS table entries needed for the file server and for the DB2 components of Natural:

FCT entry:

```
CMFSERV  DFHFCT TYPE=DATASET,                          X
               ACCMETH=VSAM ,                          X
               BUFND=5,                                X
               BUFNI=4,                                X
               DATASET=CMFSERV,                        X
               DISP=SHR,                               X
               DSNAME=SAGLIB.NCIDB2.SERVER,            X
               FILSTAT=(ENABLED,CLOSED),               X
               JID=NO,                                 X
               LOG=NO,                                 X
               LSRPOOL=NONE,  1-8 ONLY FOR XA; NONE    X
               RECFORM=(FIXED,BLOCKED),                X
               RSL=PUBLIC,                             X
               SERVREQ=(ADD,UPDATE,DELETE,BROWSE),     X
               STRNO=4
```


Step 4: Restart CICS

Restarting CICS is required, because of the additional FCT entry above.

NDB - Steps Specific to Com-plete

NDB - Steps Specific to Com-plete

Under Com-plete, the installation procedure of NDB continues with the adaptation of your Com-plete environment.

Ensure that the changes required for DB2 have been applied to your Com-plete environment (see the relevant Com-plete documentation).

NDB - Steps Specific to IMS/TM

This section describes how to install Natural for DB2 in an IMS/TM environment.

Ensure that your Natural/IMS roll buffer is large enough to contain the DB2SIZE; if you use the Natural Tools for DB2, an additional storage of 8 KB is required.

Below is information on:

- Bind Default NDB Plans for Different IMS/TM Environments
- Using Plan Selection by Using an IMS Resource Translation Table
- Using the File Server with VSAM

Bind Default NDB Plans for Different IMS/TM Environments

- JOB I055 / Steps 1631, 1632, 1633, 1634 for IMS MPP conversational, IMS BMP Natural,IMS MPP non-conversational, OBMP Natural

If desired, change library names and plan names to meet site requirements.

Using Plan Selection with IMS Resource Translation Table

If the name (or any ALIAS) of your environment-dependent Natural nucleus does not match the name of your DB2 plan, you must use an Resource Translation Table (RTT).

Below is information on:

- Step 1: Modify, assemble and link IMS RTT

Step 1: Modify, assemble and link IMS RTT

Add an additional DSNMAPN macro to your RTT as follows (for any other parameters, refer to the relevant DB2 literature by IBM):

DSNMAPN macro:

```
DSNMAPN APN=load-module,PLAN=plan-name
```

The *load-module* represents the environment-dependent Natural nucleus (that is, the IMS application program) and the *plan-name* is the same as the one used in the BIND step.

Using the File Server with VSAM

Be aware that database loops cannot be continued across terminal I/Os without using the File Server.

If you want to use the Natural File Server (VSAM), perform the following additional steps:

- Step 1: Define VSAM dataset for file server - Job I008, Step 1600
- Step 2: Format file server dataset - Job I075, Step 1600
- Step 3: Update JCL for MPP region
- Step 4: Restart your Natural/IMS MPP region

Step 1: Define VSAM dataset for file server - Job I008, Step 1600

Specify the size and the name of the VSAM RRDS that is to be used as the file server (see also Installing the File Server in Natural File Server for DB2).

Step 2: Format file server dataset - Job I075, Step 1600

Specify the five input parameters required to format the file server dataset (see also Installing the File Server in Natural File Server for DB2).

Step 3: Update JCL for MPP region

- Include the DD statement CMFSERV to define the file server dataset.
- Increase the REGION parameter if necessary.

Step 4: Restart your Natural/IMS MPP region

Restart your MPP region, because of the additional DD statement.

NDB - Steps Specific to TSO

This section describes how to install Natural for DB2 in a TSO environment:

- Using the File Server with VSAM
- Sample JCL for Starting and Using NDB under CAF
- Sample JCL for Starting and Using NDB under DSN

Using the File Server with VSAM

If you want to use the Natural File Server (VSAM), perform the following additional steps:

- Step 1: Modify NDBFSRV in NATTSO
- Step 2: Define VSAM dataset for file server - Job I008, Step 1620
- Step 3: Format file server dataset - Job I075, Step 1620

Step 1: Modify NDBFSRV in NATTSO

Set the NDBFSRV parameter in the NATTSO macro to Yes and reassemble and relink your Natural/TSO interface NATTSO.

Step 2: Define VSAM dataset for file server - Job I008, Step 1620

Specify the size and the name of the VSAM RRDS that is to be used as the file server (see also Installing the File Server in Natural File Server for DB2).

Step 3: Format file server dataset - Job I075, Step 1620

Specify the five input parameters required to format the file server dataset (see also Installing the File Server in Natural File Server for DB2).

Sample JCL for Starting and Using NDB under CAF

To test the TSO installation of NDB under CAF, perform the following steps:

- Step 1: Adapt CLIST NDBCAF - Job I070, Step 240C
- Step 2: Invoke Natural

Step 1: Adapt CLIST NDBCAF - Job I070, Step 240C

Change the library and program names in the CLIST NDBCAF to meet site requirements. If you do not use the file server, remove the ALLOC and FREE statements for CMFSERV.

Step 2: Invoke Natural

Invoke Natural by executing the CLIST created in the previous step. Ensure that DB2 tables can be accessed and that plan switching can be performed.

Before the first SQL call you must call NATPLAN to explicitly allocate the plan. The plan name must be the same as the name used in Step 4 - Create NDB plan (Installation Procedure, Steps Common to all Environments). NATPLAN can be edited to specify the appropriate DB2 subsystem ID.

Sample JCL for Starting and Using NDB under DSN

To test the TSO installation of NDB under DSN, perform the following steps:

- Step 1: Adapt CLIST NDBTSO - Job I070, Step 240B
- Step 2: Invoke Natural

Step 1: Adapt CLIST NDBTSO - Job I070, Step 240B

Change the subsystem ID as well as the library, plan and program names in the CLIST NDBTSO to meet site requirements. If you do not use the file server, remove the ALLOC and FREE statements for CMFSERV.

Step 2: Invoke Natural

Invoke Natural by executing the CLIST created in the previous step. Ensure that DB2 tables can be accessed. The plan name must be the same as the name used in the BIND step. For an explanation of the DSN and RUN commands, refer to the relevant IBM literature for DB2 TSO and batch users.

NDB - Installation Verification

This section provides example batch jobs and online methods for verifying the installation of the Natural interface to DB2:

- Test Batch NDB under CAF - Job NDBBATCA
- Test Batch NDB under DSN - Job NDBBATTB
- Test DSNMTV01 - Job NDBMTV01
- Online Verification Methods

Test Batch NDB under CAF - Job NDBBATCA

NDBBATCA contains sample JCL to test NDB in batch mode by using the CAF interface. Modify the sample JCL to meet site requirements.

Before the first SQL call you must call NATPLAN to explicitly allocate the plan. The plan name must be the same as the name used in Step 4 - Create NDB plan (Installation Procedure, Steps Common to all Environments). NATPLAN can be edited to specify the appropriate DB2 subsystem ID.

Test Batch NDB under DSN - Job NDBBATTB

NDBBATTB contains sample JCL to test NDB in batch mode by using the DSN command processor. Modify the sample JCL to meet site requirements.

The plan name must be the same as the name used in Step 4 - Create NDB plan (Installation Procedure, Steps Common to all Environments). For an explanation of the DSN and RUN commands, refer to the relevant IBM literature for DB2 TSO and batch users.

Test DSNMTV01 - Job NDBMTV01

NDBMTV01 contains a sample JCL to execute Natural by using the DB2 DL/I batch support. Modify the sample JCL to meet site requirements.

The plan name must be the same as the name used in Step 4 - Create NDB plan (Installation Procedure, Steps Common to all Environments).

Online Verification Methods

To verify the installation of the Natural interface to DB2 online, you can use either SQL Services or DEM2 example programs:

- Using SQL Services
- Using DEM2* Example Programs

Using SQL Services

To verify and check the NDB installation by using the SQL Services of the Natural SYSDDM utility

1. Invoke Natural.
2. Invoke SYSDDM.
3. On the SYSDDM main menu enter Function Code **B** to invoke the SQL Services function.
   Enter Function Code **S** and specify SQL system DB2 to select all DB2 tables.
   The communication between Natural and DB2 works if all existing DB2 tables are displayed.
   For one of the tables, generate a Natural DDM as described in Generate DDM from an SQL Table (DDM Generation).
4. After you have generated a DDM, access the corresponding DB2 table with a simple Natural program:

Example:

```
FIND view-name WITH field = value
 DISPLAY field
LOOP
END
```

If you receive the message NAT3700, enter the Natural system command SQLERR to display the corresponding SQL return code. You can find the description of the SQLERR command in the section Natural System Commands for DB2 (Natural Tools for DB2).

Using DEM2* Example Programs

To verify and test your installation you can also use the example programs DEM2* in the Natural system library SYSDB2 provided on the installation tape.

Using these example programs, you can create a DB2 table by using DEM2CREA and create the corresponding DDM via SYSDDM. You can then store data in the created table by using DEM2STOR and retrieve data from the table by using DEM2FIND or DEM2SEL. You can also drop the table by using program DEM2DROP.

Natural Parameter Modification for DB2

This section covers the following topics:

- Natural Profile Parameter Settings
- Performance Considerations for the DB2SIZE Parameter

Natural Profile Parameter Settings

To set the Natural profile parameter

1. Add the following Natural profile parameter to your NATPARM module:
   DB2SIZE=*nn*
   The DB2SIZE parameter can also be specified dynamically. It indicates the size of the DB2 buffer area, which must be set to at least 6 KB.
   The setting of DB2SIZE also depends on whether you use the file server or not. If the file server is **not** used, the setting can be calculated according to the following formula:
   ((808 + *n1**** 40 +** *n2* *** 100) + 1023) / 1024 KB**
   If the file server is used, the setting can be calculated according to the following formula:
   ((904 + *n1**** 40 +** *n2* *** 140 +***n3* *** 8) + 1023) / 1024 KB**
   The variables *n1*, *n2* and *n3* correspond to:

n1	the number of statements for dynamic access as specified as the second parameter in Job I055, Step 1600;
n2	the maximum number of nested database loops as specified with the MAXLOOP parameter in NDBPARM;
n3	the maximum number of file server blocks to be allocated per user specified as the fifth parameter in Job I075, Step 1620 or the EBPMAX parameter of NDBPARM, if you decided to use the Software AG Editor buffer pool as file server.

   Important:
   Ensure that you have also added the Natural parameters required for the Software AG Editor; see the relevant installation description in the section Installing the Software AG Editor, in the Natural Installation Guide for Mainframes).
   As DB2SIZE applies to Natural for DB2 and Natural for SQL/DS, it must be set to the maximum value if you run more than one of these environments.

2. Add an NTDB entry specifying the list of logical database numbers that relate to DB2 tables. All Natural DDMs that refer to a DB2 table must be cataloged with a DBID from this list.
   DBIDs can be any number from 1 to 254; a maximum of 254 entries can be specified. For most user environments, one entry is sufficient.
   Important:
   Ensure that all DB2 DDMs used when cataloging a given program have a valid DB2 DBID. Also ensure that the DBIDs selected in the NTDB macro for DB2 do not conflict with DBIDs selected for other database systems.
   The DBID for SQL/DS used when cataloging a Natural program does not have to be in the NTDB list of DBIDs used when executing this program. Therefore, when executing existing Natural programs, DBID 250 is not mandatory.
   Two sample NTDB macros follow:
   NTDB DB2,250
   NTDB DB2,(200,250,251)

Performance Considerations for the DB2SIZE Parameter

During execution of an SQL statement, storage is allocated dynamically to build the SQLDA for passing the host variables to DB2.

In previous Natural for DB2 versions, this storage was always obtained from the TP monitor or operating system. For performance reasons, it is now first attempted to meet the storage requirements by free space in the Natural for DB2 buffer (DB2SIZE). Only if there is not enough space available in this buffer, the TP monitor or operating system is invoked.

To take advantage of this performance enhancement, you must specify your DB2SIZE larger than calculated according to the formula (see Natural Profile Parameter Settings above).

Depending on the SQL execution mode and on the usage of the Natural file server, the additional storage requirements (in bytes) can be calculated as follows:

- Dynamic Mode
- Static Mode
- Storage Requirements for the File Server
- Sample Calculation for Dynamic Mode without Using the File Server
- Considerations for VARCHAR Fields

Dynamic Mode

- With sending fields:
  64 + *n* * 56

  With sending fields including LOB columns:
  64 + 2 * *n* * 56

  where *n* is the number of sending fields in an SQL statement.
  The storage is freed immediately after the execution of the SQL statement.

- With receiving fields (that is, with variables of the INTO list of a SELECT statement):
  64 + *n* * 56 + 24 + *n* * 2

  With receiving fields including LOB columns:
  64 + 2 * *n* * 56 + 24 + *n* * 2

  where *n* is the number of receiving fields in an SQL statement.
  The storage remains allocated until the loop is terminated.

Static Mode

- With sending fields:
  64 + *n* * 24

  With sending fields including LOB columns:
  64 + 2 * *n* * 56

  where *n* is the number of sending fields in an SQL statement.
  The storage is freed immediately after the execution of the SQL statement.

- With receiving fields (that is, with variables of the INTO list of a SELECT statement):
  64 + *n* * 24 + 24 + *n* * 2

With receiving fields including LOB columns:
64 + 2 * *n* *** 56 + 24 +** *n* *** 2**

where *n* is the number of receiving fields in an SQL statement.
The storage remains allocated until the loop is terminated.

Storage Requirements for the File Server

When using the file server, additional storage is required for each database loop that contains Positioned UPDATE and/or DELETE statements.

For each of such loops, a buffer is allocated to save the contents of all receiving fields contained in the INTO list. Therefore, the size of this buffer corresponds to the total length of all receiving fields:

20 + 4 + *sum* **(** *length* **(v1), ...,** *length* **(v***n***))**

where "v1 ... v*n*" refers to the variables contained in the INTO list.

The buffer remains allocated until the loop is terminated.

Sample Calculation for Dynamic Mode without Using the File Server

If you use the default value 10 for both variables (*n1* and *n2*), the calculated DB2SIZE will be 2208 bytes. However, if you specify a DB2SIZE of 20 KB instead, the available space for dynamically allocated storage will be 18272 bytes, which means enough space for up to either 325 sending fields or 313 receiving fields.

Since space for receiving fields remains allocated until a database loop is terminated, the number of fields that can be used inside such a loop is reduced accordingly: for example, if you retrieve 200 fields, you can update about 110 fields inside the loop.

Considerations for VARCHAR Fields

When using VARCHAR fields (that is, fields with either an accompanying L@ field in the Natural view or an explicit LINDICATOR clause), additional storage is allocated dynamically if the L@ or LINDICATOR field is not specified directly in front of the corresponding base field. Therefore, always specify these fields in front of their base fields.

NDB - Parameter Module NDBPARM

The source module NDBPARM contains Natural parameters specific to a DB2 environment. The parameter default values can be modified to meet site-specific requirements (see Step 5 in Installation Procedure, Steps Common to all Environments). The values of the parameters cannot be dynamically overwritten.

Below is a description of all NDBPARM parameters:

BTIGN	Ignore BACKOUT TRANSACTION error
CONVERS	Conversational mode under CICS
DDFSERV	Alternate DD name for NDB file server
DELIMID	Escape character for delimited identifiers
EBPFSRV	Editor buffer pool for NDB file server
EBPPRAL	Editor buffer pool primary allocation
EBPSEC	Editor buffer pool secondary allocation
EBPMAX	Editor buffer pool maximum allocation
ETIGN	Ignore END TRANSACTION error
FSERV	Activate NDB file server
MAXLOOP	Maximum nested program loops
NNPSF	Set NATURAL NUMERIC’s positive sign to F
REFRESH	Refresh setting of DB2 server and package set
RWRDONL	Generate delimited identifiers for reserved words only
STATDYN	Allow static to dynamic switch

BTIGN

Possible Values	Default Value
ON/OFF	ON

This parameter is relevant in CICS and IMS/TM environments only.

BTIGN ignores the error which results from a BACKOUT TRANSACTION statement that was issued too late for backing out the current transaction, because an implicit Syncpoint has previously been issued by the TP monitor.

Value	Explanation
ON	The error after a late BACKOUT TRANSACTION is ignored.
OFF	The error after a late BACKOUT TRANSACTION is **not** ignored.

CONVERS

Possible Values	Default Value
ON/OFF	ON

This parameter is used to allow conversational mode in CICS environments where no NDB file server is used.

Value	Explanation
ON	Conversational mode is allowed.
OFF	Conversational mode is **not** allowed.

If this parameter is set to OFF and no NDB file server is used, you cannot continue database loops across terminal I/Os; if so, the DB2 SQL codes -501, 504, 507, 514, or 518 may occur.

If you are using the SYSDDM SQL services in a CICS environment without NDB file server, you must specify CONVERS=ON, otherwise you get the errors mentioned above.

DDFSERV

Possible Values	Default Value
Any valid DD name	None

This parameter specifies a DD name for the NDB file server module other than CMFSERV.

DELIMID

Possible Values	Explanation	Default Value
"	Double quotation mark	
’	Single quotation mark	
	No value: Delimited identifiers are not enabled.	No value

This parameter determines the escape character to be used for generating delimited SQL identifiers for the column names and table names in SQL statements. A delimited identifier is a sequences of one or more characters enclosed in escape characters. You must specify a delimited identifier if you use SQL-reserved words for column names and tables names, as demonstrated in the Example of DELIMID below.

To enable generation of delimited identifiers, DELIMID must be set to double quotation mark (") or single quotation mark (’).

The escape character specified for DELIMID and the SQL STRING DELIMITER are mutually exclusive.This implies that the mark (double or single quotation) used to enclose alphanumeric strings in SQL statements must be different from the value specified for DELIMID. If you enable delimited identifiers, ensure that the value specified for DELIMID also complies with the SQL STRING DELIMITER value of your DB2 installation.

See also the RWRDONL parameter to determine which delimited identifiers are generated in the SQL string.

Example of DELIMID:

In the following example, a double quotation mark (") has been specified as the escape character for the delimited identifier:

Natural statement:

```
SELECT FUNCTION INTO #FUNCTION FROM XYZ-T1000
```

Generated SQL string:

```
SELECT "FUNCTION" FROM XYZ.T1000
```

EBPFSRV

Possible Values	Default Value
ON/OFF	ON

This parameter is used to determine whether the NDB file server uses the Software AG Editor buffer pool as storage medium or not. If it is OFF, the NDB file server uses the VSAM file as in previous versions. The parameter must be set to ON, if the NDB file server is to be used in a Sysplex environment.

EBPPRAL

Possible Values	Default Value
0 - 32676	20

This parameter specifies the number of blocks to be allocated primarily to each user of the NDB file server, if the Software AG Editor buffer pool is used as storage medium.

If the EBPFSRV parameter is set to OFF, EBPPRAL is not used at runtime.

EBPSEC

Possible Values	Default Value
0 - 32676	10

This parameter specifies the number of blocks to be allocated secondarily to each user of the NDB file server if the Software AG Editor buffer pool is used as storage medium. The secondary allocation is used to allocate buffer pool blocks to the user if the primary allocation amount is already exhausted.

If the EBPFSRV parameter is set to OFF, EBPSEC is not used at runtime.

EBPMAX

Possible Values	Default Value
0 - 32676	100

This parameter specifies the maximum number of blocks to be allocated to each user of the NDB file server if the Software AG Editor buffer pool is used as storage medium. This parameter serves as upper limit for the allocation of buffer pool blocks to a single user. If the EBPFSRV parameter is set to OFF EBPMAX is not used at runtime.

ETIGN

Possible Values	Default Value
ON/OFF	ON

This parameter is relevant in IMS/TM MPP and message-oriented BMP environments only.

It is used to handle END TRANSACTION statements in a message-driven IMS region (MPP or message-oriented BMP).

In such a region, an END TRANSACTION cannot be executed by the Natural/IMS interface and is therefore ignored without any notification. In such situations, the ETIGN parameter can be used to issue an error message instead.

Value	Explanation
ON	The END TRANSACTION error is ignored and processing is continued.
OFF	The END TRANSACTION error is **not** ignored.

FSERV

Possible Values	Default Value
ON/OFFI/DIS	OFF

This parameter determines whether the NDB file server is to be used and whether it can be disabled in the case of an initialization error.

Value	Explanation
ON	NDB file server is to be used.
OFF	NDB file server is not to be used.
DIS	NDB file server is to be used but is to be disabled if it cannot be initialized.

If FSERV is set to ON and the NDB file server is not operational, the initialization of Natural for DB2 is terminated with a corresponding Natural error message. The Natural interface to DB2 is disabled and any SQL call is rejected with a corresponding error message.

MAXLOOP

Possible Values	Default Value
1 - 99	10

This parameter specifies the maximum possible number of nested database loops.

NNPSF

Possible Values	Default Value
ON/OFF	OFF

This parameter changes the sign character of positive Natural variables which have format N, if they are filled from DB2. Usually these variables have the C as positive sign character. If the parameter NNPSF is set to ON, F is used as positive sign character.

Value	Explanation
ON	Positive numbers put into Natural NUMERIC variables by DB2 get the sign F.
OFF	Positive numbers put into Natural NUMERIC variables by DB2 get the sign as it was delivered for DB2.

REFRESH

Possible Values	Default Value
ON/OFF	OFF

This parameter is used to automatically set the DB2 server and package set to the values that applied when the last transaction was executed. Server and package set are refreshed by using the CONNECT TO *server-name* and SET CURRENT PACKAGESET = '*package-name*' SQL statements of DB2.

Value	Explanation
OFF	No automatic refresh is performed.
ON	An automatic refresh is performed every time before a database transaction starts and if a server or package set has been specified.

RWRDONL

Possible Values	Default Value
ON/OFF	ON

This parameter determines which identifiers are generated as delimited identifier in an SQL string. RWRDONL only takes effect if the setting of the DELIMID parameter allows delimited identifiers.

If RWRDONL is set to ON, only identifiers that are reserved words are generated as delimited identifiers. The list of reserved words is contained in the NDBPARM macro. This list has been merged from the lists of reserved words for DB2 for z/OS, DB2 for VSE/VM, DB2 UDB for LINUX, OS/2, Windows and UNIX, and ISO/ANSI SQL99.

If RWRDONL is set to OFF, all identifiers are generated as delimited identifiers.

STATDYN

Possible Values	Default Value
NEVER/ALWAYS/SPECIAL	NEVER

This parameter is used to allow dynamic execution of statically generated SQL statements if the static execution returns an error.

Value	Explanation
NEVER	Dynamic execution is never allowed.
ALWAYS	Dynamic execution is always allowed after an error.
SPECIAL	Dynamic execution is allowed after special errors only. These special errors are: NAT3706: Load module not found SQL -805: DBRM (database request module) does not exist in plan SQL -818: Mismatch of timestamps

Special Requirements for Natural Tools for DB2

To be able to use the Natural Tools for DB2 (see the relevant section), consider the following requirements and recommendations:

- Retrieval and Explain Functions
- LISTSQL and Explain Functions

Retrieval and Explain Functions

In order to be independent of DB2 versions, the Natural Tools for DB2 Retrieval and Explain functions have been designed not to access the DB2 catalog tables directly, but to access identical tables qualified by the creator name SYSSAG.

Thus, before you can use the Retrieval or Explain functions, you must create these tables. The SYSSAG tables must have the same columns as the DB2 catalog tables and they must be created as ALIAS, VIEW, or TABLE.

To help you create these tables, sample SQL code is provided in the member DEMSQL4 in the Natural system library SYSDB2. By default, it creates an ALIAS SYSSAG.*xxx* for the corresponding SYSIBM table.

For some catalog tables no indexes are defined. For performance reasons, consider creating copies of these tables with appropriate indexes.

For the following tables it is recommended to work with copies of the catalog tables:

SYSCOLAUTH
SYSDBRM
SYSFOREIGNKEYS
SYSINDEXPART
SYSKEYS
SYSSTMT
SYSSYNONYMS
SYSTABLEPART
SYSVIEWS

The CREATE TABLE and CREATE INDEX statements required are included as comments in the sample SQL member DEMSQL4. In addition, DEMSQLUP includes sample SQL code to update the data in the copies of the catalog tables.

For any other table, it is recommended that you create an ALIAS or a VIEW that points to the corresponding SYSIBM table.

Note:
The sample SQL members can be executed with the ISQL part of SYSDB2. ISQL enables you to read SQL members from the Natural system library SYSDB2. To save an SQL member in any other library, you can use the command LIBRARY MYLIB in the ISQL input screen to switch to another library and then save the SQL member. You cannot save SQL members in the library SYSDB2.

LISTSQL and Explain Functions

These functions access DB2 PLAN_TABLEs. To use these functions, a PLAN_TABLE must exist for your SQLID. For the layout of the PLAN_TABLE, see the relevant DB2 documentation by IBM of the EXPLAIN command.

It is recommended that you create an index on the following columns of the PLAN_TABLE:

APPLNAME
PROGNAME
COLLID
QUERYNO
TIMESTAMP DESC
QBLOCKNO
PLANNO
MIXOPSEQ

NDB - Natural for DB2 Server Stub

A Natural for DB2 (NDB) server stub is an interface module needed to communicate between the DB2 database system and the Natural server. The server stub module determines, sets up and invokes a Natural server environment for executing Natural stored procedures and Natural user-defined functions.

As mentioned in Step 11 in Installation Procedure, there are two types of server stubs: the NDB start server stub (STR) and the NDB server stub (SRV). Both stubs are generated from the NDBSTUB macro.

Below is information on:

- NDB Start Server Stub
- NDB Server Stub
- JCL Procedure
- Macro NDBSTUB

NDB Start Server Stub

The NDB start server is used for setting up the Natural server environments desired. The start server stub must be the main execution program in the Stored Procedure Address Space (SPAS). After the start server stub has established the Natural server environments, it passes control to the appropriate DB2 program (DSNX9WLM for WLM SPAS and DSNX9STP for DB2 SPAS). When SPAS terminates, the DB2 program returns control to the start server stub. The start server stub stops the Natural server environments and returns control to the operating system.

The NDB start server stub reads the names and parameters of the Natural server to be started from the CMSRVIN dataset. CMSRVIN must be specified with DDNAME CMSRVIN.

The CMSRVIN dataset is a sequential file that contains all information required to start the desired Natural servers. For each server to be started, one START entry must be provided. The parameters used for the START entries are identical to the parameters that apply to the NDBSTUB macro. Enclose the contents of each START entry in brackets and delimit comments by the following signs: /* and */.

Example of START Entries:

```
START=(SERVER=WDB41SRV,NATURAL=NATBAT4R,CMPRMIN=CMPRMIN,
        CMPRINT=CMPRINT,CMTRACE=CMTRACE,THREADSIZE=768,
        THREADNUMBER=2,TRACE=ON)
 START=(SERVER=WDB4SSRV,NATURAL=NATBAT4R,CMPRMIN=CMPRMIN,
        CMPRINT=CMPRINT,CMTRACE=CMTRACE,THREADSIZE=768,
        THREADNUMBER=2,TRACE=ON)
/*  START=(SERVER=QE41SRV,NATURAL=NATBAT41,CMPRMIN=QAPARM4, */
/*         CMPRINT=CMPRINT,CMTRACE=CMTRACE,THREADSIZE=700, */
/*         THREADNUMBER=2,TRACE=OFF) */
```

If the start server dataset is missing or has not been assigned, the start server stub will start a Natural server environment with the parameters that derive from the parameters defined for the start server stub itself.

NDB Server Stub

The NDB server stub is the link between DB2 and Natural stored procedures or Natural user-defined functions (Natural UDFs). Specify the NDB server stub as EXTERNAL NAME in the SYSIBM.SYSROUTINES table row that refers to the Natural stored procedure or Natural UDF. The server stub is started by DB2/WLM when the Natural stored procedures or Natural UDFs are invoked. The NDB server stub creates a Natural session in the

Natural server environment and invokes the Natural subprogram comprising the Natural stored procedure or the Natural UDF.

A Natural session created for executing a Natural stored procedure terminates when the corresponding Natural subprogram ends and control returns to DB2 and to the calling client.

A Natural session created for executing a Natural UDF stays active for multiple function invocations if the PARALLEL attribute is set to **D** and the FINAL CALL attribute is set to **Y**. The session invoked for a Natural UDF function is terminated by the server stub if it detects a termination call.

JCL Procedure

The JCL procedure of the Stored Procedure Address Space (SPAS) must specify the NDB start server stub as program in the EXEC statement.

The NDB start server stub and the NDB server stub must reside in a library contained in the steplib concatenation of the JCL procedure of the SPAS.

Example JCL:

```
//************************************************************
//*    JCL FOR RUNNING THE WLM-ESTABLISHED STORED PROCEDURES
//*    ADDRESS SPACE
//*       RGN     -- MVS REGION SIZE FOR THE ADDRESS SPACE.
//*       DB2SSN  -- DB2 SUBSYSTEM NAME.
//*       NUMTCB  -- NUMBER OF TCBS USED TO
//*                  PROCESS END USER REQUESTS.
//*       APPLENV -- MVS WLM APPLICATION ENVIRONMENT
//*                  SUPPORTED BY THIS JCL PROCEDURE.
//*
//************************************************************
//DB27ENV2 PROC RGN=0K,APPLENV=DB27ENV2,DB2SSN=DB27,NUMTCB=8
//IEFPROC EXEC PGM=WDB41STR,REGION=&RGN,TIME=NOLIMIT,  /*  NDB start server stub
//*IEFPROC EXEC PGM=DSNX9WLM,REGION=&RGN,TIME=NOLIMIT,
//        PARM='&DB2SSN,&NUMTCB,&APPLENV'
//STEPLIB  DD  DISP=SHR,DSN=DSN710.RUNLIB.LOAD
//         DD  DISP=SHR,DSN=CEE.SCEERUN
//         DD  DISP=SHR,DSN=DSN710.SDSNLOAD
//         DD  DISP=SHR,DSN=NATURAL.V2.TEST.NUCLEUS    /* Library containing stubs and Natural nucleus
//CMPRMIN  DD  DISP=SHR,DSN=SAG.SYSF.SOURCE2(TDB31PRM) /* Dynamic Natural parameters.
//CMSRVIN  DD  DISP=SHR,DSN=SAG.SYSF.SOURCE2(CMSRVIN)  /* Servers to be started.
//CEEDUMP  DD  SYSOUT=X
//SYSOUT   DD  SYSOUT=X                                /* Traces records of the NDB server stub
//RMTRACE  DD  SYSOUT=X
//CMPRINT  DD  SYSOUT=X
//SYSPRINT DD  SYSOUT=X
//SYSERROR DD  SYSOUT=X
//SYSUDUMP DD  SYSOUT=X
```

Macro NDBSTUB

The NDBSTUB macro is used to generate the NDB server stub and NDB start server stub. You can parameterize NDBSTUB to create different stubs.

Below are the parameters available with NDBSTUB:

CMPRINT	DDNAME of MPRINT dataset
CMPRMIN	DDNAME of CMPRMIN dataset
CMTRACE	DDNAME of CMTRACE dataset
GTRACE	NDB server stub to execute GTRACE calls
GTRCID	GTRACE ID to be used
MAIN	No longer relevant and only maintained for compatibility reasons
MODE	Operation mode of NDB server stub
NATURAL	Name of server front-end or Natural server
SERVER	Server name for Natural server environment
THREADNUMBER	No longer relevant and only maintained for compatibility reasons
THREADSIZE	Size of Natural threads for Natural server
TRACE	NDB server stub to write trace records
WLM	NDB start server stub mode WLM/DB2 SPAS

CMPRINT

Possible Values	Default Value
8 character DDNAME	CMPRINT

CMPRINT specifies the DDNAME of the CMPRINT dataset to which the primary report output is written. If an asterisk (*) is specified, a unique DDNAME P*nnnnnnn* is built whenever a Natural stored procedure is invoked.

CMPRMIN

Possible Values	Default Value
8 character DDNAME	CMPRMIN

CMPRMIN specifies the DDNAME of the CMPRMIN dataset during startup to read the input PROFILE parameter for this server.

CMTRACE

Possible Values	Default Value
8 character DDNAME	CMTRACE

CMTRACE specifies the DDNAME of the CMTRACE dataset to which the primary report output is written. If an asterisk (*) is specified, a unique DDNAME P*nnnnnnn* is built whenever a Natural stored procedure is invoked which makes it possible to store each output separately.

GTRACE

GTRACE specifies whether the NDB server stub executes GTRACE macro calls.

Possible Values	Default Value
ON/OFF	OFF

GTRACE specifies whether or not the server stub executes GTRACE macro calls for tracing purposes.

Value	Explanation
ON	The generated server stub executes GTRACE macros in order to document its processing.
OFF	The generated server stub does not execute GTRACE macros during its processing cycle.

GTRCID

Possible Values	Default Value
Decimal number from 0 to 1023	203

GTRCID specifies the event ID recorded with the trace data created by the NDB server stub.

MAIN

Possible Values	Default Value
YES/NO	YES

The value of MAIN is no longer evaluated. The NDB server stubs check whether they are invoked as IBM LE (Language Environment) main program or as IBM LE subprograms and react accordingly.

Value	Explanation
YES	The generated server stub operates as IBM Language Environment main program.
NO	The generated server stub operates as IBM Language Environment sub program.

MODE

Possible Values	Default Value
STR/SRV	SRV

MODE determines the operational mode of the NDB server stub generated.

Value	Explanation
STR	The generated NDB server stub operates as NDB start server stub that sets up the Natural server environment.
SRV	The generated NDB server stub operates as NDB server stub that invokes the associated Natural stored procedure or Natural UDF.

NATURAL

Possible Values	Default Value
Any valid load module name	NATBAT41

NATURAL denotes the name of the server front-end or Natural server load module which will be loaded by the NDB server stub if the external CMSTART is not already resolved by the linkage editor during creation of the server stub. The named load module has to be present in any steplib of the stored procedure address space.

SERVER

Possible Values	Default Value
Up to 5 characters	NDB41

Server names suffixed with the three characters SRV denote the names of the servers used by the server front-end in order to identify the Natural server. These names must be unique within one address space.

THREADNUMBER

Possible Values	Default Value
Decimal number	10

The value of THREADNUMBER is no longer evaluated. Instead, the NDB start server stub uses the NUMTCB parameter of the SPAS JCL procedure as THREADNUMBER value. For further details, see the relevant DB2 literature by IBM.

THREADNUMBER determines the number of Natural threads used by the Natural server. This number limits the number of Natural stored procedures and Natural UDFs concurrently active in the Natural server.

THREADSIZE

Possible Values	Default Value
Decimal number	768

THREADSIZE determines the size of the Natural threads to be used by the Natural server. The size is specified in units of kilobytes.

TRACE

Determines whether the NDB server stub generated writes trace records. The trace records are written to the dataset specified with DDNAME SYSOUT.

Possible Values	Default Value
YES/NO	NO

WLM

WLM (Workload Manager) specifies where control is passed to after the NDB start server stub has established the Natural server environments requested. This parameter is only evaluated if the MODE parameter is set to MODE=STR. Specify WLM=YES if the NDB start server stub runs in an address space that has been established by WLM.

Value	Explanation
YES	The start server stub generated links to DSNX9WLM, after setting up the Natural server environments.
NO	The start server stub generated links to DSNX9STP, after setting up the Natural server environments. WLM=NO is the default.

Natural Tools for DB2 - Overview

Using Natural Tools for DB2	Invoke the Natural Tools for DB2. Edit within the Natural Tools for DB2. Gobal PF-key settings. Global maintenance commands.
Application Plan Maintenance	Maintain application plans and packages.
Catalog Maintenance	Maintain the DB2 catalog.
Procedure Maintenance	Define, list and insert DB2 stored procedures.
Interactive SQL	Generate and issue interactive SQL statements.
Retrieval of System Tables	Retrieve data from system tables.
Environment Setting	Display and modify environment settings.
Explain PLAN_TABLE	List and Explain PLAN_TABLEs.
File Server Statistics	Display file server statistics.
Issuing DB2 Commands from Natural	Issue DB2 commands from Natural.
Natural System Commands for DB2	Display and explain generated SQL statements.
Natural Tools for DB2 with Natural Security	Restrict the use of the Natural Tools for DB by Natural Security.

Using Natural Tools for DB2

This section covers the following topics:

- Invoking Natural Tools for DB2
- Editing within the Natural Tools for DB2
- Global PF-Key Settings
- Global Maintenance Commands

Invoking Natural Tools for DB2

To invoke Natural Tools for DB2

- In the Natural command line, enter the system command SYSDB2.

The Natural Tools for DB2 Main Menu is displayed, which offers you the following functions:

Main Menu - Functions

Application Plan Maintenance	Maintain DB2 application plans online.
Catalog Maintenance	Maintain the DB2 catalog.
Procedure Maintenance	Insert and maintain DB2 stored procedures.
Interactive SQL	Process SQL statements that are not embedded.
Retrieval of System Tables	Display/print DB2 objects and user authorizations.
Environment Setting	Execute SQL statements and display special register values.
Explain PLAN_TABLE	Interpret your PLAN_TABLE.
File Server Statistics	Display statistics on the generation and use of the file server.
DB2 Commands Execution	Issue DB2 commands from Natural.

Note:
If you have created a new SYSDB2 library when installing Natural for DB2, ensure that it contains all Predict interface programs necessary to run the Natural Tools for DB2. These programs are loaded into SYSDB2 at Predict installation time (see the relevant Predict documentation).

Editing within the Natural Tools for DB2

The free-form editor available within the Natural Tools for DB2 requires that the Software AG Editor is installed. The main and line commands available for use within the Natural Tools for DB2 are a subset of those available within this editor.

Main commands are entered in the command line of the editor screen. The most important main commands are:

Command	PF Key	Description
BOTTOM (++)		Positions to the bottom of the data.
CHANGE		Scans for a specified string and replaces each such string found with another specified string.
CLEAR		Clears the editor source area.
DELETE		Deletes the line(s) containing a given string according to the specified selection operands.
DOWN (+)	PF 8	Scrolls the specified scroll amount downwards.
FIND		Finds a string specified by command operands at the location(s) specified by selection operands.
LEFT	PF 10	Scrolls the specified scroll amount to the left.
LIMIT *n*		Sets a limit for the FIND command; *n* lines are processed.
PRINT		Prints the data displayed.
RESET		Resets all pending line commands.
RFIND	PF 5	Repeats the last FIND command.
RIGHT	PF 11	Scrolls the specified scroll amount to the right.
TOP (--)		Positions to the top of the data.
UP (-)	PF 7	Scrolls the specified scroll amount upwards.

The scroll amount for the UP, DOWN, LEFT, and RIGHT commands is specified in the SCROLL field at the top right corner of the list screen. Valid values for the scroll amount are:

Value	Explanation
CSR	Scroll amount is determined by cursor position
DATA	Scroll amount equals the page size less one line
HALF	Scroll amount is half the page size
MAX	Scroll amount equals the amount of data to the bottom/top
PAGE	Scroll amount is equal to the page size
n	Scroll amount is equal to *n* lines

Line commands are entered in the editor prefix area of the corresponding statement line. The most important line commands are:

Command	Description
A	Inserts line(s) to be moved or copied after the current line.
B	Inserts line(s) to be moved or copied before the current line.
C	Copies the current line.
CC	Marks the beginning and end of a block of lines to be copied.
D	Deletes the current line.
DD	Marks the beginning and end of a block of lines to be deleted.
I*nn*	Inserts *nn* new lines after the current line.
M	Moves the current line.
MM	Marks the beginning and end of a block of lines to be moved.

Both main commands and line commands are described in detail as part of the Natural Tools for DB2 online help facility, which is invoked by pressing PF1 (Help). For further details, please refer to the Software AG Editor documentation.

Global PF-Key Settings

Within the Natural Tools for DB2, the following global PF-key settings apply:

Key	Setting	Description
PF1	Help	Pressing PF1 invokes the Natural Tools for DB2 online help system from any screen within the Natural Tools for DB2.
PF3	Exit	Pressing PF3 always takes you to the previous screen or function. When pressed on an editor screen where modifications have been made, the Exit Function window is displayed (as described in Exit Function in Program Editor and Data Area Editor in the Natural Editors documentation). When you press PF3 on the Main Menu, you leave the Natural Tools for DB2.
PF12	Canc	Pressing PF12 always takes you back to the menu from where the current screen has been invoked. When you press PF12 on the Main Menu, you leave the Natural Tools for DB2.

Global Maintenance Commands

Within the Natural Tools for DB2, the following global maintenance commands apply:

Command	Description
COPY *name*	Copies the specified member from the current library into the editor, after (A) or before (B) the current line.
LIBRARY *name*	Specifies the Natural library "*name*" as current library.
LIST *name**	Lists all members from the current library whose names start with "*name*". From the list, you can select a member by marking it with "S".
PURGE *name*	Purges the specified member from the current library.
READ *name*	Reads the specified member from the current library into the editor. The current name is set to "*name*".
SAVE [*name*]	Saves the generated code as the member *"name"* in the current library. If no name is specified, the current name is taken. Current library and member names are displayed above the command line.

Member and library names must correspond to the Natural naming conventions: see Object Names (section: Editors - General Information, Getting Started) and "Naming Conventions for Libraries" in the section LOGON (Natural System Command Reference documentation). Members can be JCL members, SQL members, or output members.

NDB - Application Plan Maintenance

The application plan maintenance part of the Natural Tools for DB2 is used to generate JCL code to:

- create database request modules (DBRMs) from your Natural programs,
- maintain DB2 application plans and packages from within your Natural environment.

Two modes of operation are available: fixed mode and free mode.

In fixed mode, maintenance screens with syntax graphs help you to specify the correct commands. Complete JCL members can be generated using predefined job profiles. You simply enter the required data in input maps. The data are checked to ensure that they comply with the correct syntax. Then JCL members are generated from these data. The members can be submitted directly by pressing PF4 (Submi). But you can also switch to free mode by pressing PF5 (Free).

Pressing PF5 in fixed mode invokes the free-mode editor, which can be used to modify JCL code generated in fixed mode, without the syntactical restrictions imposed. In free mode you can submit the JCL member currently in the source area by pressing PF4 (as in fixed mode).

This section covers the following topics:

- Commands and PF-Key Settings
- Invoking the Application Plan Maintenance Function
- Prepare Job Profiles
- Create DBRMs
- Bind Plan
- Rebind Plan
- Free Plan
- Bind Package
- Rebind Package
- Free Package
- List JCL Function
- Display Job Output

Commands and PF-Key Settings

Within the maintenance screens in fixed mode, various windows can be invoked. These windows are accessed via 1-byte control fields.

To invoke such a window

- Enter "S" in the corresponding control field.

If the control field displays an "X", data have already been entered in the corresponding window.

In addition, the following PF-key settings apply in fixed mode:

Key	Function
PF4	Generates JCL code and submits it.
PF5	Generates JCL code and enters free mode.
PF6	Scrolls to the top of a window.
PF7	Scrolls backwards in windows.
PF8	Scrolls forwards in windows.
PF9	Scrolls to the bottom of a window.
PF10	Shows the previous screen.
PF11	Shows the next screen.

In free mode, JCL code can be edited and submitted. Editing of JCL code is done via edit and line commands; see Editing within the Natural Tools for DB2.

Generated JCL code is submitted by pressing PF4 (Submi).

Apart from being submitted, JCL code can also be copied, listed, purged, retrieved from, or saved in a Natural library. All this is done via maintenance commands.

Invoking the Application Plan Maintenance Function

To invoke the Application Plan Maintenance function

- Enter function code "A" on the Natural Tools for DB2 Main Menu.

The Application Plan Maintenance menu is displayed:

```
16:14:02              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
                       - Application Plan Maintenance -

                  Code Function              Parameter

                   PP  Prepare Job Profile
                   CD  Create DBRMs          Lib
                   BI  Bind                  Lib, Obj
                   RB  Rebind                Lib, Obj
                   FR  Free                  Lib, Obj
                   LJ  List   JCL            Lib, JCL
                   JO  Display Job Output    Node
                   ?   Help
                   .   Exit

            Code .. __  Object ...... ________
                        Library ..... SAG_____
                        JCL Member .. ________
                        Node ........ 148

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                    Logn        Canc
```

The following functions are available:

Code	Description
PP	Defines job profiles for DBRM creation and plan/package maintenance.
CD	Generates JCL to create database request modules.
BI	Generates JCL to bind a plan or package.
RB	Generates JCL to rebind a plan or package.
FR	Generates JCL to free a plan or package.
LJ	Invokes the free-mode editor.
JO	Displays job output. This function only applies if the Entire System Server is installed.

In addition, four parameters are available, which must be specified according to the selected function:

Parameter	Description
Object	Specifies whether to maintain a plan ("PLAN" or "PL") or a package ("PACKAGE" or "PK").
Library	Specifies the name of a Natural source library. All existing libraries except the ones beginning with "SYS" can be specified; a library must be specified for JCL maintenance. The library name is preset with your Natural user ID.
JCL Member	If a valid member name is specified, the corresponding JCL member is displayed. If a value is specified followed by an asterisk (*), all JCL members in the specified library whose names begin with this value are listed. If asterisk notation is specified only, a selection list of all JCL members in the specified library is displayed. If the JCL Member field is left blank, the empty free-mode editor screen is displayed.
Node	Specifies the number of the node to be used by the Entire System Server. The default number "148" can be overwritten.

Prepare Job Profiles

If you want to generate JCL to create a DBRM or to bind, free, or rebind a plan or package, you have to specify a job name, job cards, and the name of a job profile. Thus, you have to prepare the job profiles first. Once your job profiles are defined, you can always immediately select the corresponding function if you want to create a new DBRM or if you want to bind, free, or rebind an a plan or package using your predefined job profiles.

To define a job profile

- Invoke the Prepare Job Profile function by entering function code "PP" on the Application Plan Maintenance menu.

The Prepare Job Profile menu is invoked:

```
 16:14:33               ***** NATURAL TOOLS FOR DB2 *****           1999-09-30
                             - Prepare Job Profile -




                        Code Function

                         J   Default Job Cards
                         D   Profile for Create DBRM Job
                         P   Profile for DSN Jobs
                         ?   Help
                         .   Exit

                  Code .. _   Profile .. ________

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit                                                   Canc
```

Code	Description
J	Defines user-specific default job cards.
D	Defines job profiles for the DBRM creation function.
P	Defines job profiles for the plan or package maintenance functions.

In addition, the parameter Profile is available, which is relevant to function codes "D" and "P" only. With function code "J", Profile corresponds to "USER".

Parameter	Description
Profile	Specifies the name of an already existing job profile. If a valid profile name is specified, the free-mode editor with the specified job profile is invoked, where the profile can be modified and saved. If a value is specified followed by an asterisk (*), all existing job profiles whose names begin with this value are listed. If asterisk notation is specified only, a selection list of all existing job profiles is displayed. If the field is left blank, the corresponding fixed-mode profile screen is invoked, where a new job profile can be created. To save the new profile, you have to switch to free mode.

Job profiles can be maintained (that is, copied, listed, purged, retrieved from, or saved in a Natural library) via maintenance commands.

Note:
Job profiles are saved on the Natural system file (FNAT).

Default Job Cards

All jobs generated by the Application Plan Maintenance function require job cards. With the Default Job Cards function, you can define a default job card for each user. The default job cards apply to all function screens on which you can generate JCL. Default job cards can be invoked and modified on all these screens. Asterisk notation (*) can be used to select the desired job card from a list.

To define a default job card, invoke the Default Job Cards function by entering function code "J" in the Prepare Job Profile menu. The Default Job Cards screen is invoked, where you can create and save your user-specific job cards. To do so, you can also read (directly or from a list) and modify an already existing default job card. Existing job cards can be purged, too.

```
16:14:33               ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
                              - Default Job Cards -




 R ead,  S ave,  L ist or  P urge  Default Job Cards _      User ID .. ________

   Job Name ... ________
   Job Cards ..
     //          JOB ____________________________________________________________
     // ____________________________________________________________________________
     // ____________________________________________________________________________
     // ____________________________________________________________________________
     // ____________________________________________________________________________


Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                   Canc
```

As you will see on the following pages, all function screens used to specify jobs contain the same two fields - Job Name and Job Cards - as the Default Job Cards screen. Thus, it is possible to override the default job cards in each of these screens, too.

The Job Name field enables you to change the name of the job.

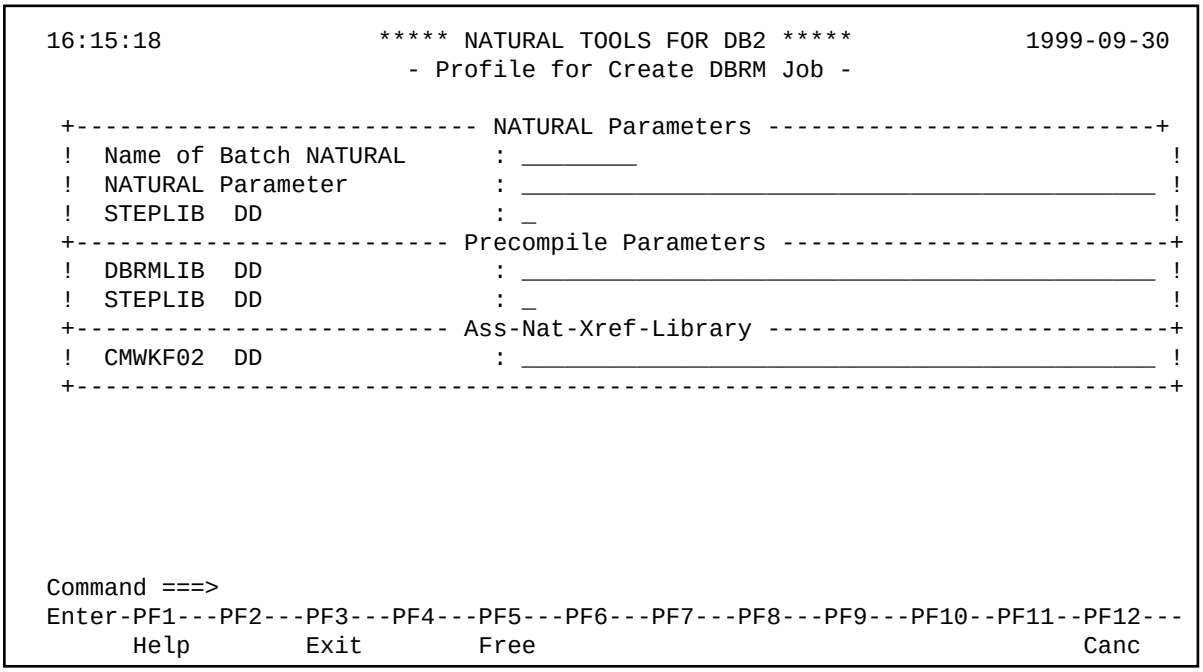
In the Job Cards field, you can enter an "S" to invoke a window where you can modify all the job cards.

Profile for Create DBRM Job

The Profile for Create DBRM Job function enables you to define profiles for the Create DBRMs functions. Job profiles for DBRM creation consist of JCL which includes the following predefined set of substitution parameters:

Parameter	Description
@JOBCARDS	Is replaced by the job cards entered on the create DBRM screen (up to five lines). You can also code the job cards in the profile and omit the job cards modifier.
@COMMAND	Is replaced by the string "CREATE DBRM".
@DBRMNAME	Is replaced by the name of the DBRM, which can be up to eight characters long.
@CREATE-DBRM	Is replaced by the command input for the static generation step. This parameter must be placed **after** the //CMSYNIN card and must comply with the Assembler naming conventions.
@COMMAN2	Is replaced by the string "MODIFY".
@MODIFY	Is replaced by the command input for the static modification step.
@XR-START @XR-END	Both mark the JCL to contain the Natural Assembler XREF data; if no XREF option is specified, the JCL is deleted again.

To define a job profile for DBRM creation, invoke the Profile for Create DBRM Job function by entering function code "D" on the Prepare Job Profile menu. If you specify a valid profile name, too, the free-mode editor containing the specified profile is invoked, where you can modify, save, and rename the displayed profile. If you leave the Profile field blank, the Profile for Create DBRM Job screen is invoked, which helps you in creating a new profile. To save the newly created job profile, you have to switch to free mode by pressing PF5 (Free).



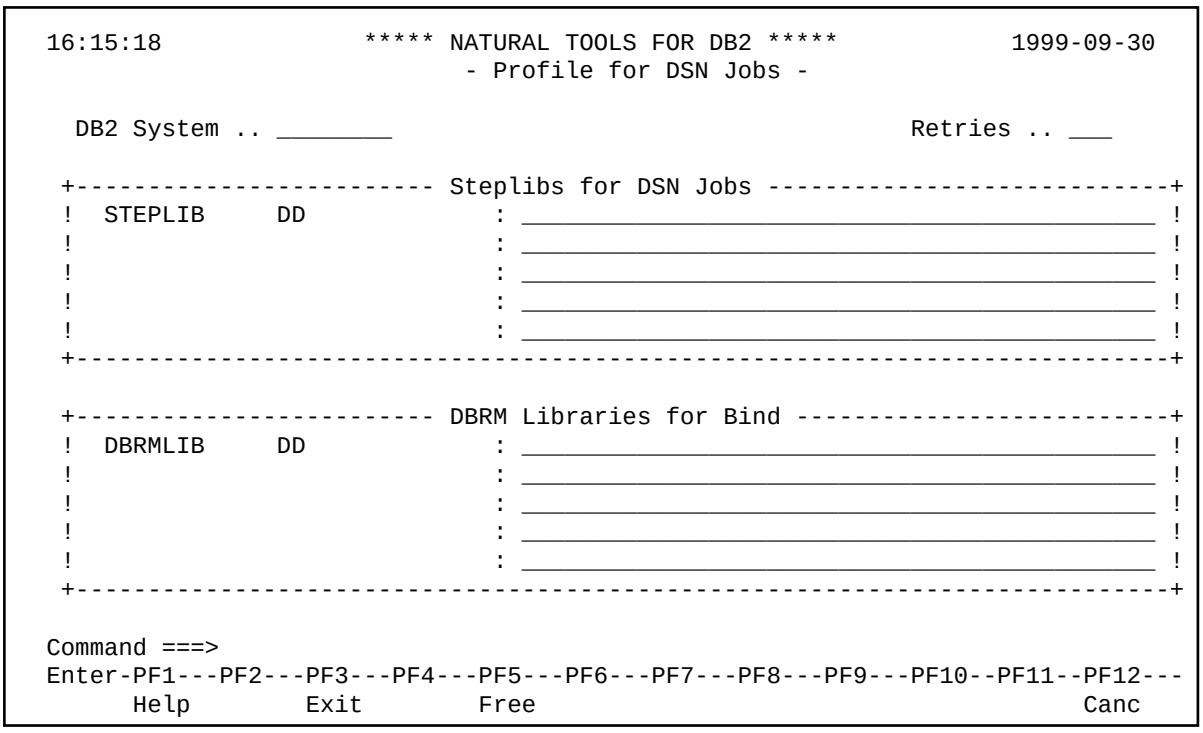
Profile for DSN Jobs

The Profile for DSN Jobs function enables you to define profiles for the Bind, Rebind, and Free functions. The same profiles can be used for each of the three functions.

Profiles for DSN jobs consist of JCL which includes the following predefined set of substitution parameters:

Parameter	Description
@JOBCARDS	Is replaced by the current job cards; you can also code the job cards in the profile and omit the job cards modifier.
@DSNCMD	Is replaced by the command input for the bind, rebind or free function.
@PLANNAME	For the bind function, it is replaced by the name of the plan or package. For the rebind and free functions, it is set to blank.
@COMMAND	Is replaced by the string "BIND", "REBIND" or "FREE" respectively.

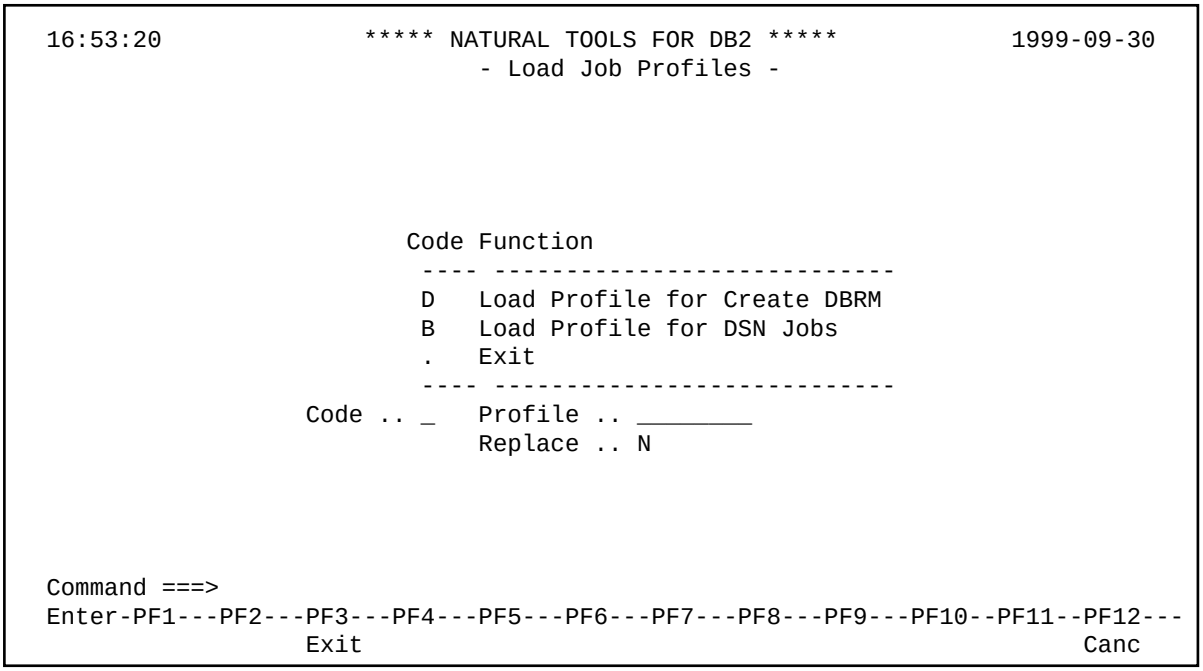
To define a profile for DSN jobs, invoke the Profile for DSN Jobs function by entering function code "P" on the Prepare Job Profile menu. If, in addition, you specify a valid profile name, the free-mode editor containing the specified profile is invoked, where you can modify, save, and rename the displayed profile. If you leave the Profile field blank, the Profile for DSN Jobs screen is invoked, which helps you in creating a new profile for DSN jobs. To save the newly created job profile, you have to switch to free mode by pressing PF5 (Free).



Loading Job Profiles

Job profiles for DBRM creation and plan/package maintenance are loaded from the dataset CMWKF01 in batch mode using the LOADPROF command.

LOADPROF is issued in the Natural system library SYSDB2; the following menu is displayed:



The following functions are available:

Code	Description
D	Serves to load job profiles for DBRM creation.
B	Serves to load job profiles for plan or package maintenance.

The following parameters apply:

Parameter	Description
Profile	Specifies the name of the profile to be loaded. This parameter must be specified.
Replace	Specifies whether it is to be replaced or not if a profile with the specified name already exists. Y An already existing profile is replaced. N An already existing profile is **not** replaced. This parameter is optional; the default setting is "N".

Unloading Job Profiles

Job profiles for DBRM creation and plan/package maintenance are unloaded and written to the dataset CMWKF01 in batch mode using the UNLDPROF command.

If UNLDPROF is issued in the Natural system library SYSDB2, the following menu is displayed:

```
16:53:20              ***** NATURAL TOOLS FOR DB2 *****           1999-09-30
                              Load Job Profiles




                        Code Function

                         D   Unload Profile for Create DBRM
                         B   Unload Profile for DSN Jobs
                         .   Exit

                  Code .. _   Profile .. ________




Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                   Exit                                                   Canc
```

The following functions are available:

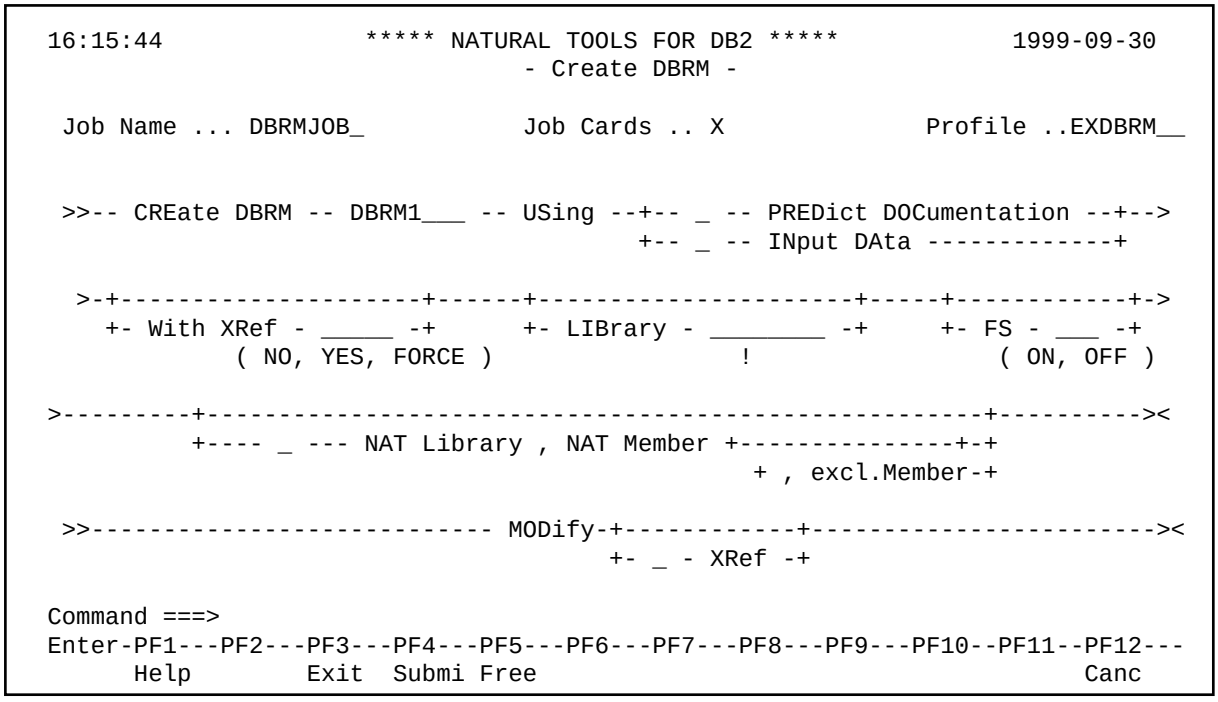
Code	Description
D	Unloads job profiles for DBRM creation.
B	Unloads job profiles for plan or package maintenance.

The following parameter applies:

Parameter	Description
Profile	Specifies the name of the profile to be unloaded. This parameter must be specified.

Create DBRMs

To create a DBRM you have to generate JCL for DBRM creation. To do so, invoke the Create DBRMs function by entering function code "CD" on the Application Plan Maintenance menu. The Create DBRM screen is invoked, where, in addition to a job name, your user-specific default job cards, and the desired job profile, you can specify all necessary information for the CREATE DBRM and MODIFY commands



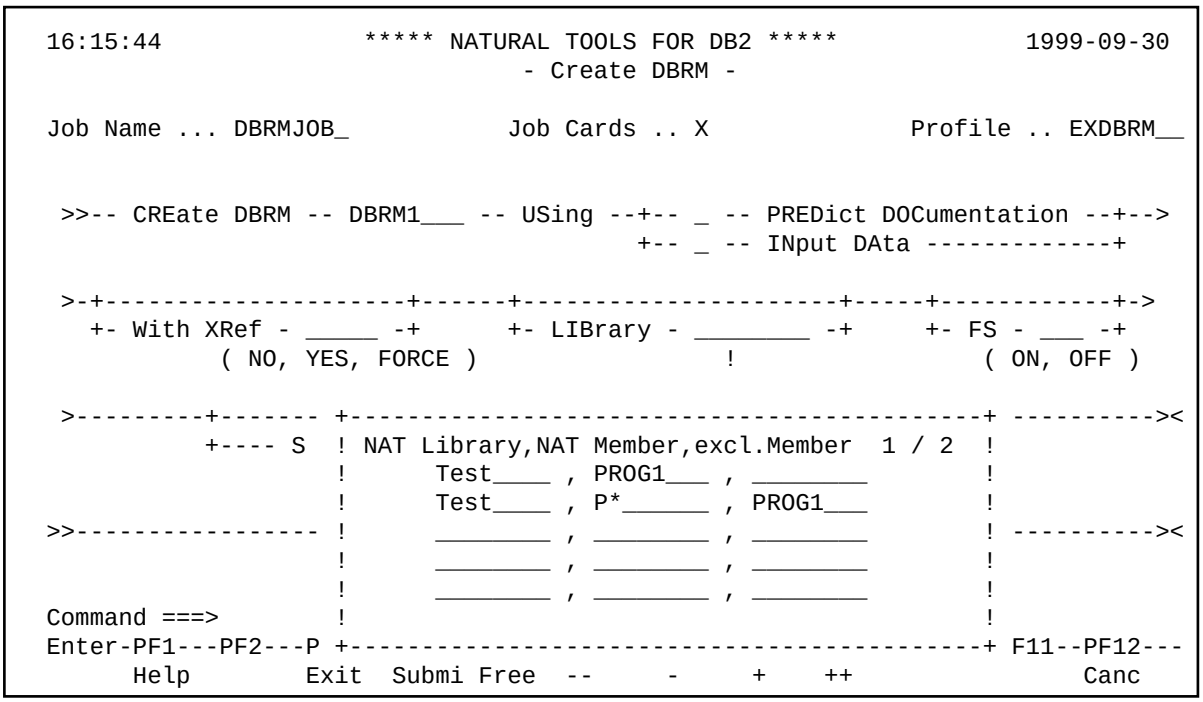
In the Job Name field, a valid job name must be specified. If you only want to change the name of the job, you can do this using the Job Name field, too.

Via the Job Cards field, you can override your default job cards. To do so, enter an "S" in the Job Cards field. A window containing your job cards is displayed.

An "X" in the Job Cards field indicates that job cards for DBRM creation are defined. A blank Job Cards field indicates that no job cards are defined.

In the Profile field, you can specify the name of a valid job profile for DBRM creation. If a value is specified followed by an asterisk (*), all existing job profiles whose names begin with this value are listed. If asterisk notation is specified only, a selection list of all available job profiles is displayed.

If you use the INPUT DATA option, a window is displayed, where you have to specify the Natural libraries and programs (members) to be contained in the DBRM.



In the third column of the above window, you can specify a program that is to be excluded from the DBRM; this is possible only if you specify an asterisk with the program name in the second column.

Within the window, you can scroll using PF6 (--), PF7 (-), PF8 (+), or PF9 (++).

The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free), or submitted immediately by pressing PF4 (Submi).

Bind Plan

To generate JCL to bind a plan, invoke the Bind function by entering function code "BI" on the Application Plan Maintenance menu and "PLAN" or "PL" in the Object field. The first Bind Plan screen is invoked, where all necessary information must be specified.

```
23:16:38              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
                                 - Bind Plan -

 Job Name ... BINDJOB_           Job Cards .. X             Profile .. EXBIND1_


 >>- BIND +-----+------------+-+------------------+-------------------------+>
          !                  ! !                  ! !                        !
          + PLAN ( TESTPLAN )+ + OWNER ( ________ )+ + QUALIFIER ( ________ )+
                   plan-name              auth-id               qualifier-name

  >-+->-- MEMBER +- X ---(member name)---+---------------------------------+-+
    !                                    !                                 ! !
    !                                    +- LIBRARY -- _ --(library name)-+ !
    !                                                                      !
    +->-- PKLIST -- X --(+------+---------+collection-id.package-id)--------+->
                         +-location-name.-+

     Read member name/package list from PREDICT?  N (Y/N)  DONE

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Submi Free                               Next  Canc
```

Apart from the specifications to be made in the Job Name, Job Cards, and Profile fields, to BIND a plan, you have to specify the name of the plan and all DBRMs and/or packages that are to be bound into the specified plan.

You invoke the window to specify the DBRM members and/or package lists by entering an "S" in the MEMBER and/or PKLIST field respectively. Either or both windows must be invoked; otherwise, you are prompted by the system to do so.

Within the windows for DBRM and package specification, you can scroll using PF6 (--), PF7 (-), PF8 (+), or PF9 (++).

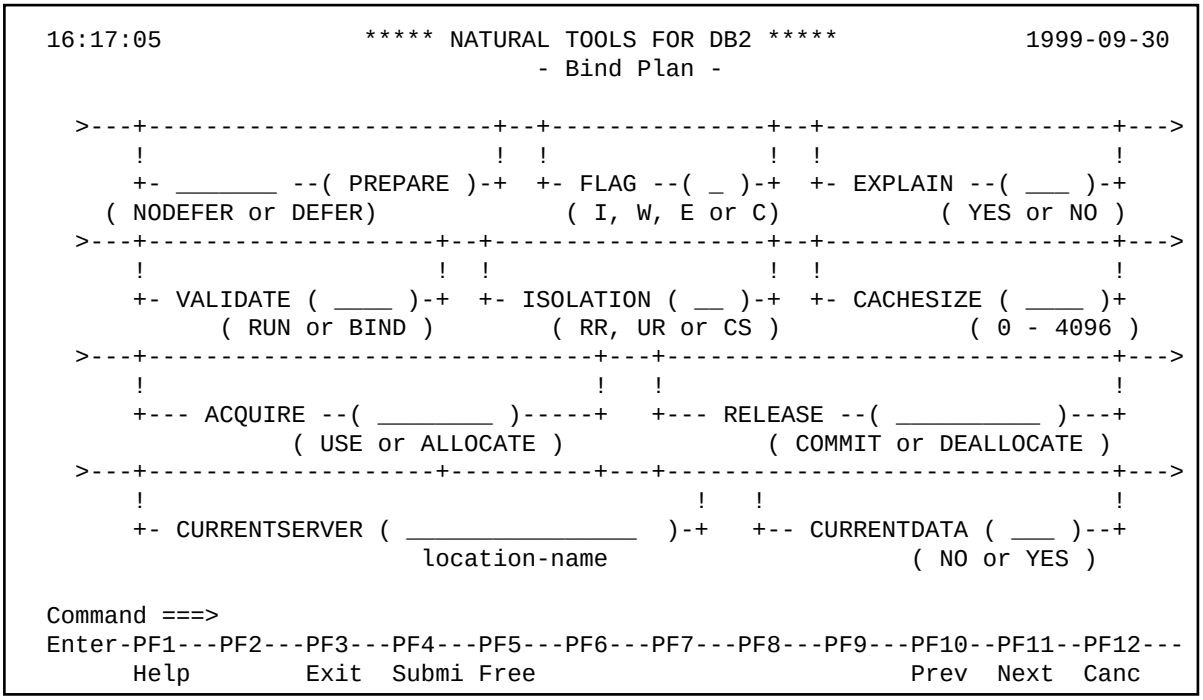
If Predict is installed and a plan is documented in Predict, the DBRM members and/or package lists assigned to a plan in Predict can be read by entering "Y" for this option (default is "N"). A maximum of 50 DBRM members and/or 20 package lists can be read.

If you use this option and DBRM members and/or package lists have been successfully read, the MEMBER and PKLIST selection fields are marked with "X", and "DONE" is displayed next to the "(Y/N)" input field; "FAILED" is displayed if:

- inconsistencies in the member/package list definition were detected,
- over 50 DBRM members or more than 20 package lists were defined for the specified plan,
- no members or package lists were defined for the specified plan,
- the plan was not documented in Predict at all.

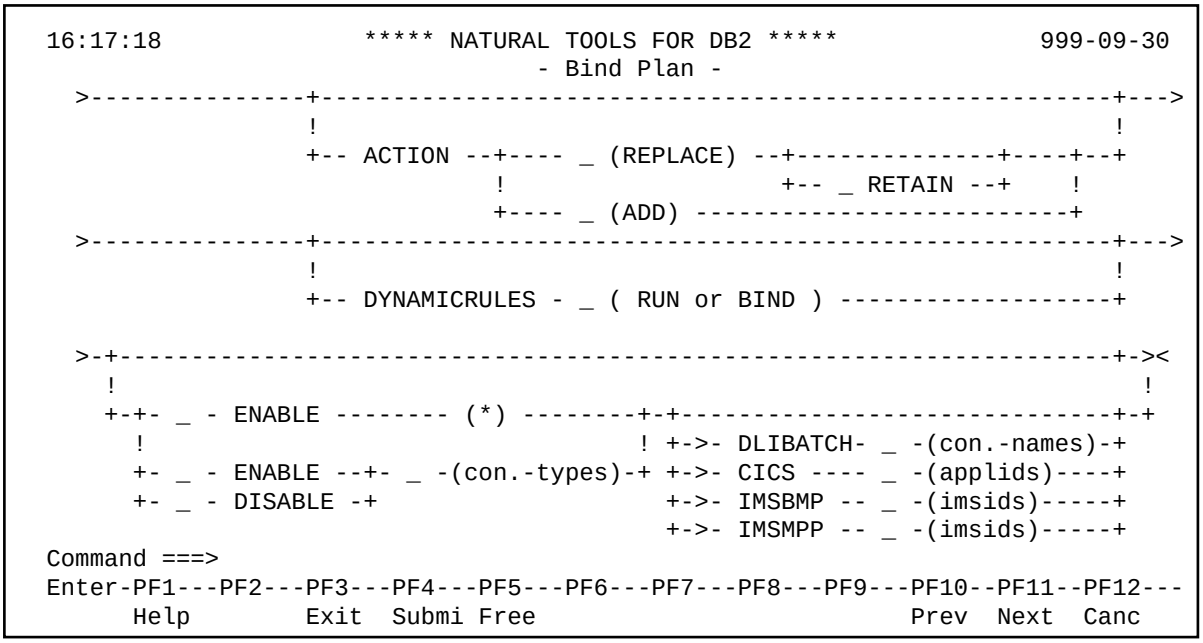
Note:
If Predict is not installed, the field "Read member name / package list from Predict?" does not appear on the above screen.

Pressing PF11 (Next) takes you to a second Bind Plan screen, where you can specify further options of the DB2 BIND command. A keyword is generated by entering its first letter in the corresponding input field; the default values are highlighted.

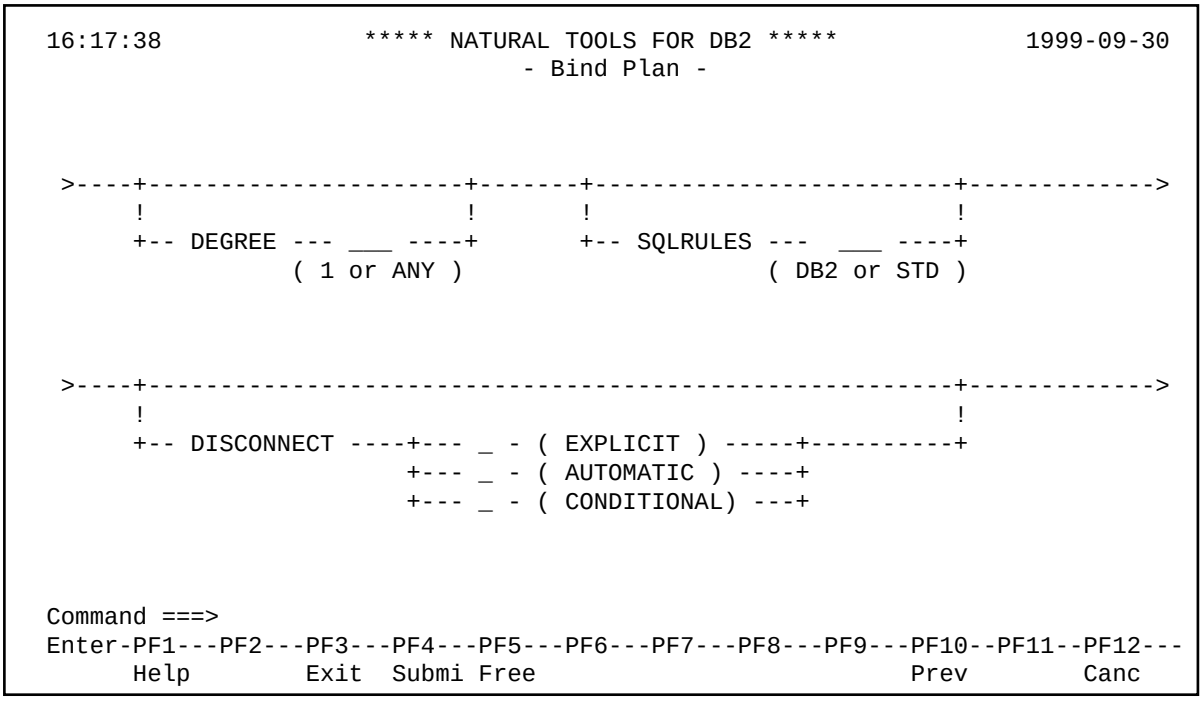


Pressing PF10 (Prev) takes you back to the previous screen.

Pressing PF11 (Next) takes you to a third Bind Plan screen, where you can again specify further options of the DB2 BIND command.



Pressing PF11 (Next) takes you to a fourth Bind Plan screen, where you can again specify further options of the DB2 BIND command.

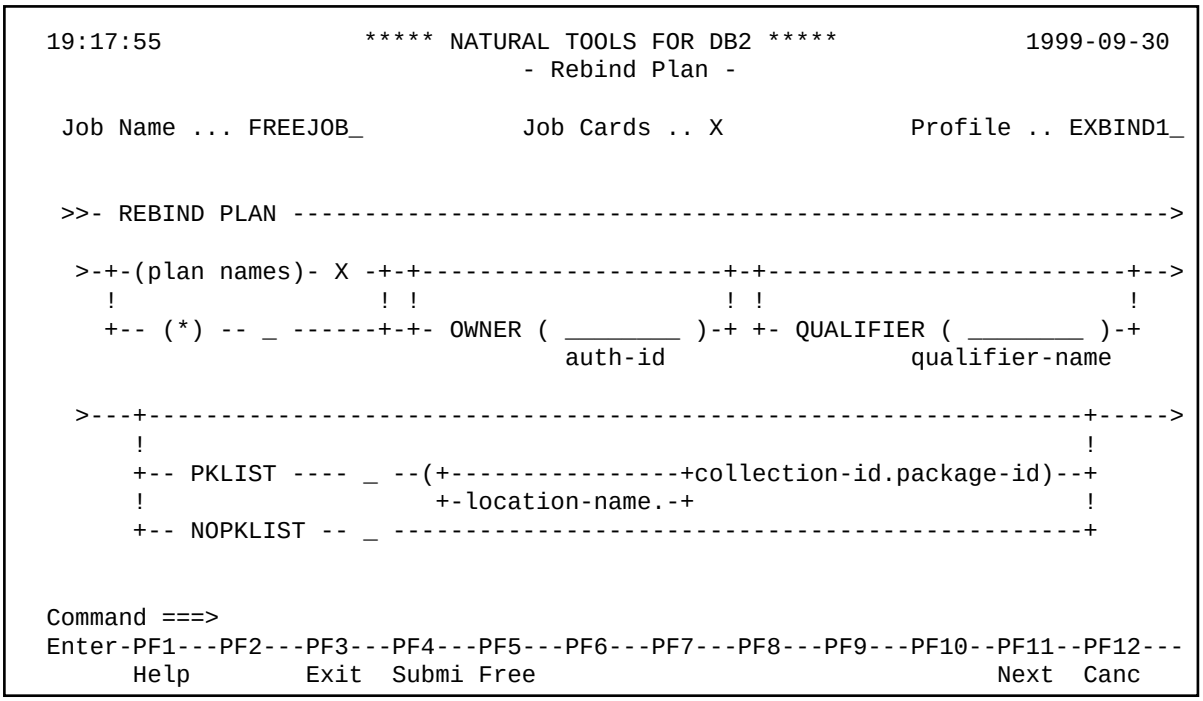


All parameters necessary to bind a plan are entered on these four screens, which show the syntax of the DB2 BIND PLAN command.

The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free), or submitted immediately by pressing PF4 (Submi).

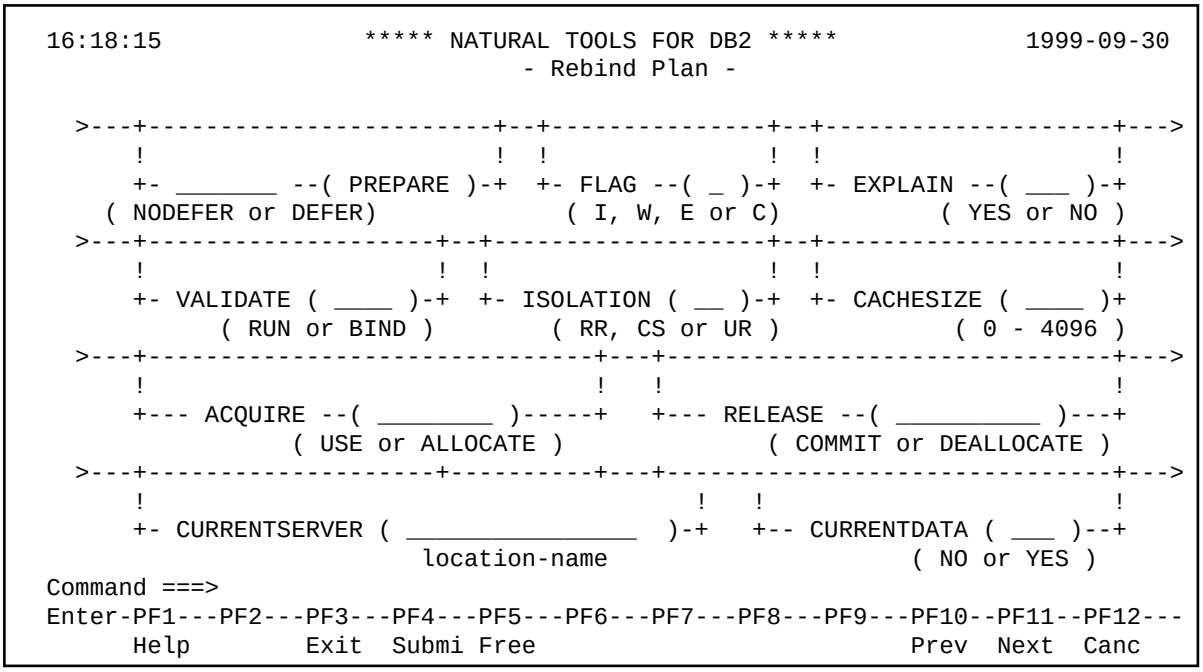
Rebind Plan

To generate JCL to rebind a plan, invoke the Rebind function by entering function code "RB" on the Application Plan Maintenance menu and "PLAN" or "PL" in the Object field. The first Rebind Plan screen is invoked, where all necessary information must be specified.



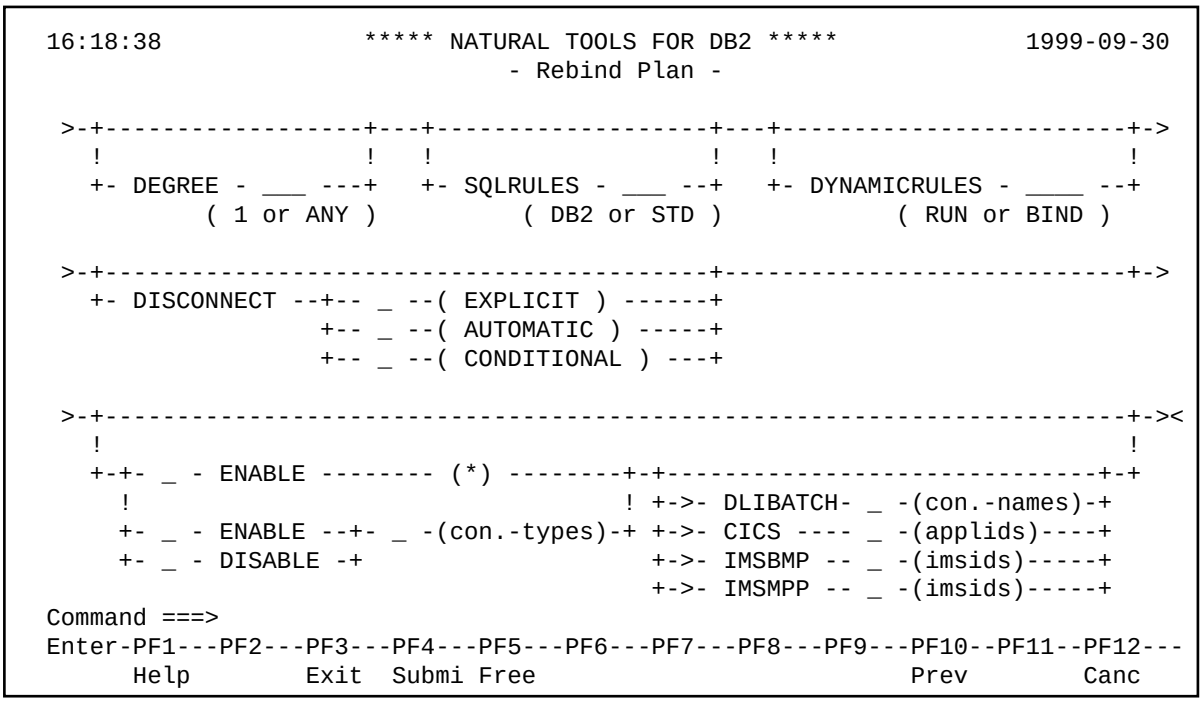
Apart from the specifications to be made in the Job Name, Job Cards, and Profile fields, you have to specify the names of the plans to be rebound in a window. If you specify asterisk notation (*), all existing plans are rebound.

Pressing PF11 (Next) takes you to a second Rebind Plan screen, where you can specify further options of the DB2 REBIND command. A keyword is generated by entering its first letter in the corresponding input field; the default values are highlighted.



Pressing PF10 (Prev) takes you back to the previous screen.

Pressing PF11 (Next) takes you to a third Rebind Plan screen, where you can again specify further options of the DB2 REBIND command.

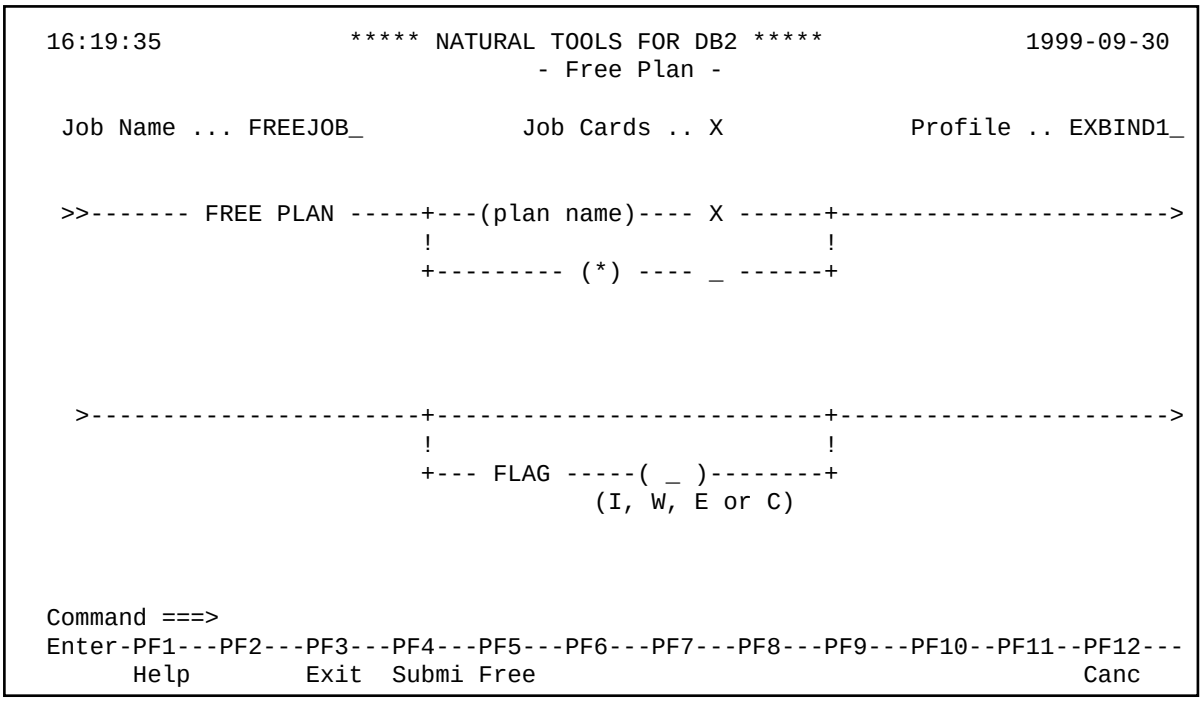


All parameters necessary to rebind a plan are entered in these three screens, which show the syntax of the DB2 REBIND PLAN command.

The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free), or submitted immediately by pressing PF4 (Submi).

Free Plan

To generate JCL to free a plan, invoke the Free function by entering function code "FR" on the Application Plan Maintenance menu and "PLAN" or "PL" in the Object field. The Free Plan screen is invoked, where all necessary information must be specified.

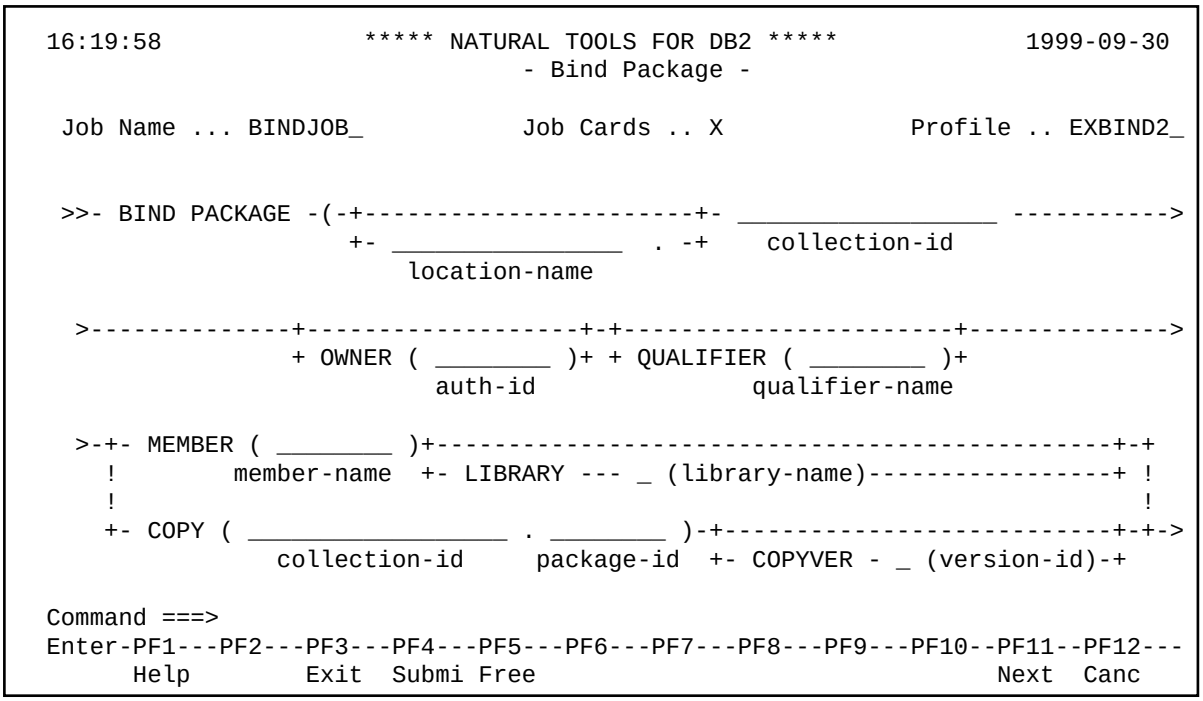


Apart from the specifications to be made in the Job Name, Job Cards, and Profile fields, all parameters necessary to free a plan are entered in a screen showing the syntax of the DB2 FREE PLAN command. The names of the plans to be freed are entered in a window. If you specify asterisk notation (*), all plans are freed.

The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free), or submitted immediately by pressing PF4 (Submi).

Bind Package

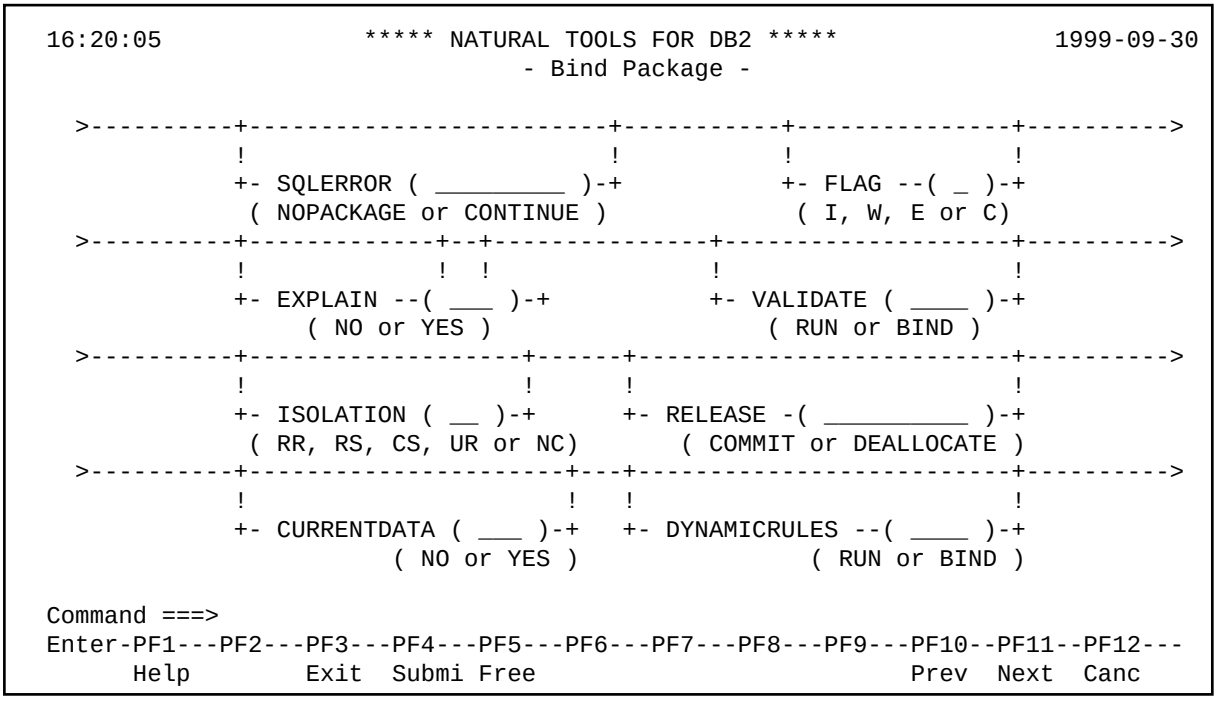
To generate JCL to bind a package, invoke the Bind function by entering function code "BI" on the Application Plan Maintenance menu and "PACKAGE" or "PK" in the Object field. The first Bind Package screen is invoked, where all necessary information must be specified.



Apart from the specifications to be made in the Job Name, Job Cards, and Profile fields, to BIND a package, you have to specify the collection ID of the package and a DBRM or a further package to be bound into the specified package.

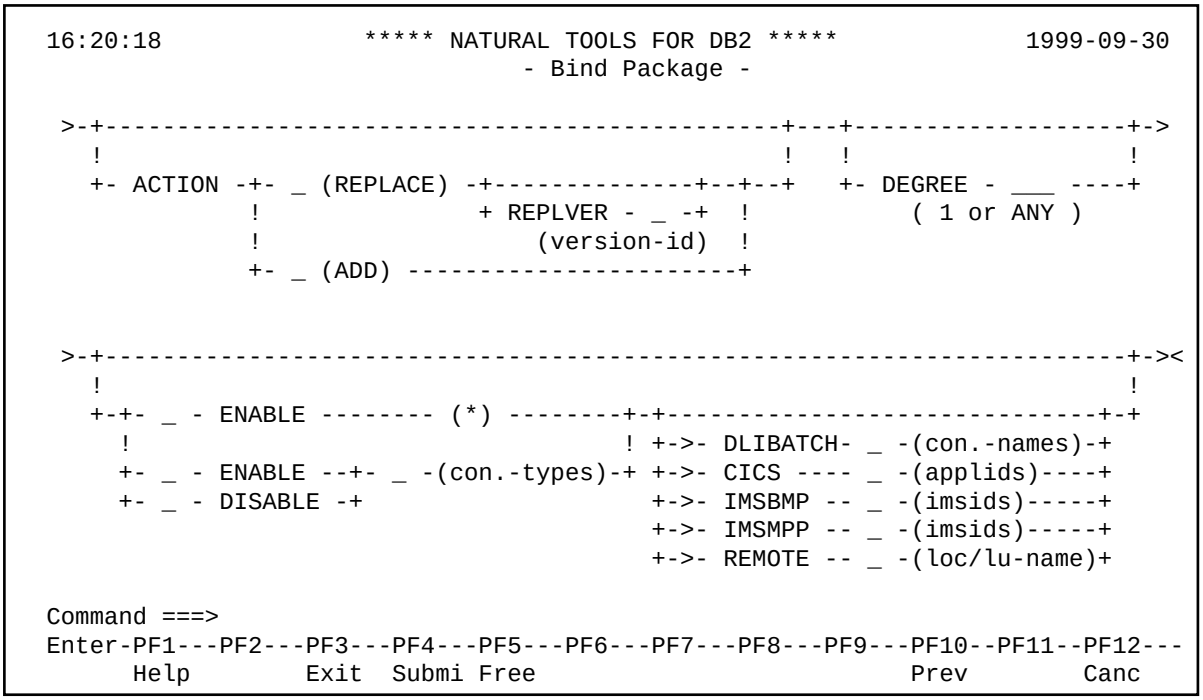
You specify the DBRM or the second package in the MEMBER or COPY field respectively. Either of the fields must be selected and the package ID will be either the DBRM name or the package ID of the copied package.

Pressing PF11 (Next) takes you to a second Bind Package screen, where you can specify further options of the DB2 BIND command. A keyword is generated by entering its first letter in the corresponding input field; the default values are highlighted.



Pressing PF10 (Prev) takes you back to the previous screen.

Pressing PF11 (Next) takes you to a third Bind Package screen, where you can again specify further options of the DB2 BIND command.



All parameters necessary to bind a package are entered on these three screens, which show the syntax of the DB2 BIND package command.

The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free), or submitted immediately by pressing PF4 (Submi).

Rebind Package

To generate JCL to rebind a package, invoke the Rebind function by entering function code "RB" on the Application Plan Maintenance menu and "PACKAGE" or "PK" in the Object field. The first Rebind Package screen is invoked, where all necessary information must be specified.

```
16:20:55              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
                                - Rebind Package -

 Job Name ... FREEJOB_           Job Cards .. X             Profile .. EXBIND2_


 >>- REBIND PACKAGE ------------------------------------------------------------>


  >-+--- _ -------------------------- (*) ----------------------------------+->
    !                                                                        !
    +--- _ -(+----------------+-collection-id.package-id-+---------------+)-+
             +-location-name.-+                          +-.(version-id)-+


  >------------+---------------------+-+-------------------------+----------->
               !                     ! !                         !
               +- OWNER ( ________ )-+ +- QUALIFIER ( ________ )-+
                          auth-id                 qualifier-name
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Submi Free                                 Next  Canc
```

Apart from the specifications to be made in the Job Name, Job Cards, and Profile fields, you have to specify the names of the packages to be rebound in a window. If you specify asterisk notation (*), all locally existing packages are rebound.

Pressing PF11 (Next) takes you to a second Rebind package screen, where you can specify further options of the DB2 REBIND command. A keyword is generated by entering its first letter in the corresponding input field; the default values are highlighted.


```
16:21:21              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
                                - Rebind Package -

  >----------+--------------------+-----------+--------------------+--------->
             !                    !           !                    !
             +- FLAG -----( _ )---+           +- DEGREE --( ___ )--+
                  ( I, W, E or C)                    ( 1 or ANY )
  >----------+--------------------+-----------+--------------------+--------->
             !                    !           !                    !
             +- EXPLAIN --( ___ )-+           +- VALIDATE ( ____ )-+
                  ( NO or YES )                   ( RUN or BIND )
  >----------+--------------------+-----+-----------------------+----------->
             !                    !     !                       !
             +- ISOLATION ( __ )--+     +- RELEASE -( _________ )-+
             ( RR, RS, CS, UR or NC )      ( COMMIT or DEALLOCATE )
  >----------+---------------------+---+-------------------------+---------->
             !                     !   !                         !
             +- CURRENTDATA ( ___ )-+   +- DYNAMICRULES -( ____ )--+
                       ( NO or YES )               ( RUN OR BIND )

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Submi Free                          Prev  Next  Canc
```

Pressing PF10 (Prev) takes you back to the previous screen.

Pressing PF11 (Next) takes you to a third Rebind package screen, where you can again specify further options of the DB2 REBIND command.

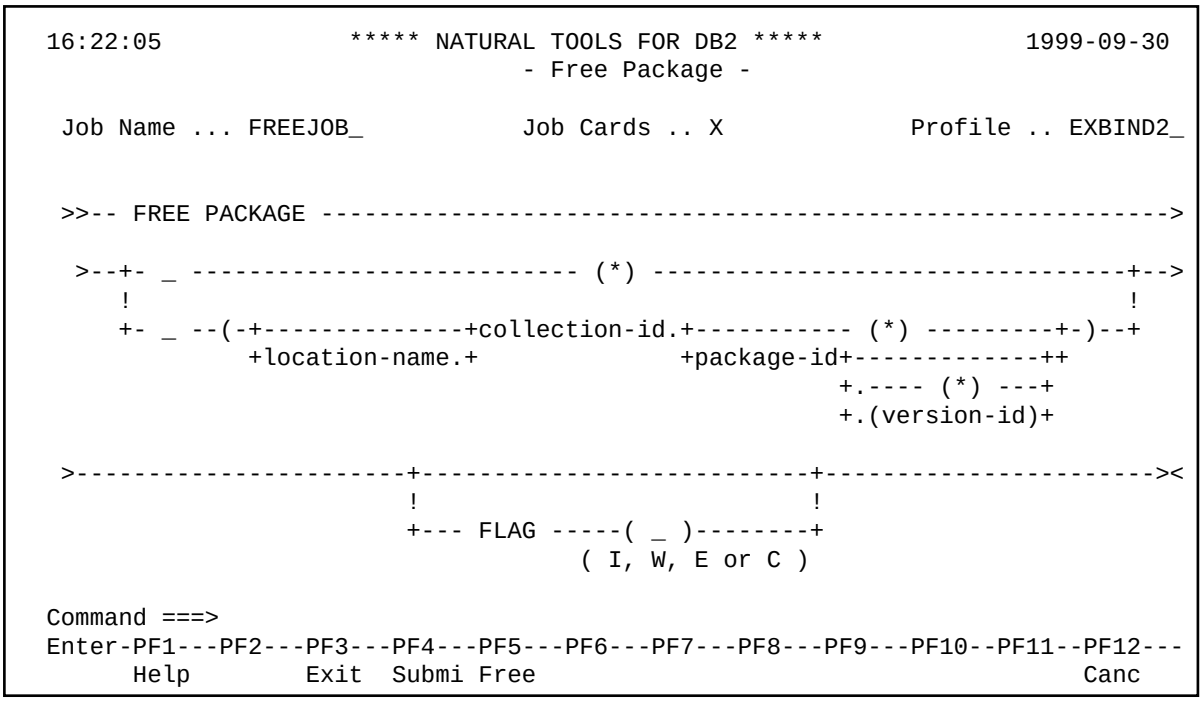
```
16:21:38              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
                                - Rebind Package -




 >-+-------------------------------------------------------------------------+-><
   !                                                                         !
   +-+- _ - ENABLE -------- (*) --------+-+-----------------------------+-+
     !                                  ! +->- DLIBATCH- _ -(con.-names)-+
     +- _ - ENABLE --+- _ -(con.-types)-+ +->- CICS ---- _ -(applids)----+
     +- _ - DISABLE -+                    +->- IMSBMP -- _ -(imsids)-----+
                                          +->- IMSMPP -- _ -(imsids)-----+
                                          +->- REMOTE -- _ -(loc/lu-name)+




Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Submi Free                          Prev        Canc
```

All parameters necessary to rebind a package are entered in these two screens, which show the syntax of the DB2 REBIND PACKAGE command

The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free), or submitted immediately by pressing PF4 (Submi).

Free Package

To generate JCL to free a package, invoke the Free Package function by entering function code "FR" on the Application Plan Maintenance menu and "PACKAGE" or "PK" in the Object field. The Free Package screen is invoked, where all necessary information must be specified.



Apart from the specifications to be made in the Job Name, Job Cards, and Profile fields, all parameters necessary to free a package are entered in a screen showing the syntax of the DB2 FREE PACKAGE command. The names of the packages to be freed are entered in a window. If you specify asterisk notation (*), all local packages are freed.

The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free), or submitted immediately by pressing PF4 (Submi).

List JCL Function

The List JCL function serves to invoke the free-mode editor via the Application Plan Maintenance menu. To do so, enter function code "LJ". If you leave the JCL Member field blank, the empty free-mode editor is invoked. If you specify a value followed by an asterisk, or specify asterisk notation only, a list of JCL members is displayed for selection. If you specify a valid member name, the invoked free-mode editor contains the corresponding JCL.

```
16:18:18              ***** NATURAL TOOLS FOR DB2 *****              1999-09-30
 APM - free mode     TESTLIB(TESTPLAN)     S 01- ---------------Columns 001 072
=====>                                                          Scroll ===>  PAGE
 ***** ***************************** top of data ****************************
 00001 //BINDJOB  JOB TESTPLAN,CLASS=K,MSGCLASS=X
 00002 //*************************************************************
 00003 //* EXAMPLE JOB PROFILE FOR BIND, FREE AND REBIND           *
 00004 //*                                                         *
 00005 //* BIND PLAN                                               *
 00006 //*************************************************************
 00007 //BINDJOB  EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=4096K
 00008 //STEPLIB  DD DSN=DB2.Vnnn.DSNLOAD,DISP=SHR
 00009 //DBRMLIB  DD DSN=DB2.Vnnn.DBRMLIB.DATA,DISP=SHR
 00010 //SYSTSPRT DD SYSOUT=*
 00011 //SYSPRINT DD SYSOUT=*
 00012 //SYSUDUMP DD SYSOUT=*
 00013 //SYSTSIN  DD *
 00014 DSN SYSTEM (DB2)
 00015     BIND PLAN (PLAN1)
 00016     MEMBER (  DBRM1)
 00017 END

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Submi Rfind Rchan -     +           <      >   Canc
```

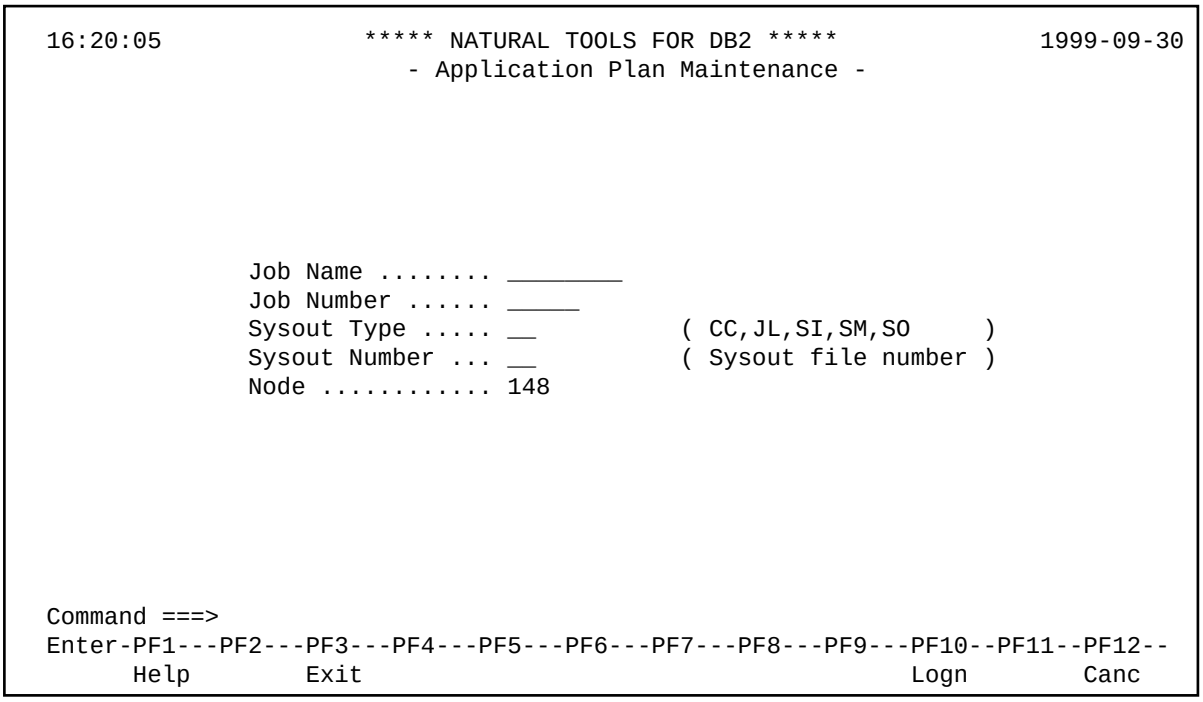
Within the free-mode editor, JCL members can be copied, listed, purged, retrieved from, or saved in a Natural library. All this is done via maintenance commands.

Press PF4 (Submi) to submit JCL code listed in the editor, press PF5 (Fix) to switch to fixed mode.

Display Job Output

The Display Job Output function is available only if the Entire System Server is installed.

If you want to display the output of a JCL member, enter function code "JO" on the Application Plan Maintenance menu to invoke the Display Job Output function; the default node number (148) for Entire System Server can be modified. A screen is invoked, where you can specify the desired job name and job number, as well as the numbers of the SYSOUT types.



In the Job Name field, a valid job name can be specified.

If you specify a value followed by an asterisk, or specify asterisk notation only, a list of job output members is displayed for selection. In a job output member selection list, you can mark an output member with either "B" to display the member only, or "L" to display a list of all the job output's SYSOUTs, which in turn can be marked with "B" for display.

If you leave the Job Name field blank, you must specify a job number.

In the Job Number field, you can specify a unique job number. Only if a unique job number has been specified, specifications can be made in the Sysout Type and Sysout Number fields, too.

In the Sysout Type field, you can specify the type of SYSOUT dataset of the job with the specified job number to be displayed. The following codes apply:

Code	SYSOUT Type
CC	Condition Code
JL	Job Listing
SI	System Input
SM	System Message
SO	System Output

In the Sysout Number field, you can specify a file number to display a specific SYSOUT dataset of the type specified in the Sysout type field. If you leave the Sysout Number field blank, all SYSOUT datasets of the specified type are displayed.

NDB - Catalog Maintenance

The Catalog Maintenance part of the Natural Tools for DB2 enables you to generate SQL statements to maintain the DB2 catalog (that is, DB2 tables and other DB2 objects) without leaving your development environment.

The Catalog Maintenance function incorporates an SQL generator that automatically generates from your input the SQL code required to maintain the desired DB2 object. You can display, modify, save, and retrieve the generated SQL code.

The DDL/TML definitions are stored in the current Natural library.

This section covers the following topics:

- Fixed Mode and Free Mode
- Invoking the Catalog Maintenance Function
- Create Table Function
- Create Tablespace Function
- Alter Table Function
- Alter Tablespace Function
- SQL Skeleton Members

Fixed Mode and Free Mode

The catalog maintenance function offers two modes of operation: Fixed Mode and Free Mode. To switch from fixed mode to free mode, you press PF5. If you press PF3 (Exit) in free mode, you are returned to fixed mode.

Fixed Mode

In fixed mode, input screens with syntax graphs help you to specify correct SQL code. You simply enter the required data in the input screens, and the data are automatically checked to ensure that they comply with the DB2 SQL syntax. If the input is incomplete, you are prompted for the missing data. Then, SQL members are generated from the entered data. The members can be executed directly by pressing PF4. But you can also switch to free mode, where the generated SQL code can be modified.

After the execution of an SQL statement, a message is returned, which indicates that the statement has been successfully executed. If an error occurred, the resulting DB2 error message can be displayed by pressing PF2 (Error), which executes the SQLERR command.

Input screens consist of various kinds of input fields. There are:

- fields to enter DB2 object names,
- fields to invoke windows,
- fields to be marked for selection,
- fields to enter keywords,
- fields to specify numeric values,
- fields to enter string constants.

For each field where a window can be invoked, you can specify an "S". When you press ENTER, the window appears and you can select or enter the necessary information. If such a selection is required, an "S" is already preset when the corresponding screen is invoked.

When you press ENTER again, the window closes and if data have been entered, the field is marked with "X" instead of "S". If not, the field is left blank or marked with "S" again.

This will continue each time you press ENTER until no "S" remains. To redisplay a window where data have been entered, you change its "X" mark back to "S".

If another letter or character is used, an error message appears on the screen:

Mark field with ’S’ to show window.

The wrong character is automatically replaced by an "S" and if you press ENTER again, the corresponding window appears.

In fields where keywords are to be entered, you must enter one of the keywords displayed beneath the field. Default keywords are highlighted.

Free Mode

When free mode is invoked from fixed mode, the data that were entered in fixed mode are shown as generated SQL code which can be saved for later use or modification.

If you modify an SQL member in free mode, this has no effect on the fixed-mode version of the member. You can save your modified code in free mode, but when you return to fixed mode, the original data appear again. Thus, both original and modified data are available.

In free mode you can execute the member currently in the source area by pressing PF4 (as in fixed mode).

Execution of SQL statements automatically switches to the output screen, which shows the SQL return code of the executed commands.

See the list of the SQL code maintenance commands available in free mode.

Invoking the Catalog Maintenance Function

To invoke the Catalog Maintenance function

- Enter function code "C" on the Natural Tools for DB2 Main Menu.

The Catalog Maintenance menu is displayed:


```
16:03:13              ***** NATURAL TOOLS FOR DB2 *****               1999-09-30
                             - Catalog Maintenance -

      Code   Maintenance  Parameter        Code   Authorization  Parameter

       CR    CREATE       Object            GR    GRANT          Object
       AL    ALTER        Object            RE    REVOKE         Object
       DR    DROP                           LO    LOCK TABLE
       SC    SET SQLID

      Code   Description  Parameter        Code   Function       Parameter

       EN    EXPLAIN                        F     Free Mode      Member
       CO    COMMENT ON                     ?     Help
       LB    LABEL ON                       .     Exit

   Code .. __  Object .... ________
               Library ... ________
               Member .... ________


Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                  Canc
```

In the Code field, the function code assigned to the desired function can be specified, together with the desired Object, Library, and/or Member name.

If you switch to free mode and enter a valid member name, you can read this member from the Natural library specified with the Library parameter. The Library parameter is preset with your Natural user ID.

With the CREATE VIEW and EXPLAIN functions, a subselect or an explainable SQL statement must be entered, respectively. Both can be done in a separate editor session, where previously saved members can be used. The editor is invoked by entering an "S" in the appropriate field.

With the functions CREATE, ALTER, GRANT, and REVOKE, an object code must be specified, for example, "TB" for TABLE. If you leave the object field blank, a window is displayed which shows you a list of all available objects together with their object codes.

If you enter for example the CREATE function without specifying an object, a window is invoked which prompts you for the type of object to be created:


```
16:03:13              ***** NATURAL TOOLS FOR DB2 *****               1999-09-30
                             - Catalog Maintenance -

      Code  +----------------------+       Code   Authorization  Parameter
            ! CREATE               !
       CR   !                      !        GR    GRANT          Object
       AL   ! AL    ALIAS          !        RE    REVOKE         Object
       DR   ! DB    DATABASE       !        LO    LOCK TABLE
       SC   ! IX    INDEX          !
            ! ST    STOGROUP       !
      Code  ! SY    SYNONYM        !       Code   Function       Parameter
            ! TB    TABLE          !
       EN   ! TS    TABLESPACE     !        F     Free Mode      Member
       CO   ! VI    VIEW           !        ?     Help
       LB   ! .     Exit           !        .     Exit
            !                      !
   Code ..  ! __ .. Enter Object   !
            !                      !
            +----------------------+


Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                 Canc
```

In the following section some examples illustrate how to use the catalog maintenance function in fixed mode.

Create Table Function

If you enter the object code "TB" in the CREATE function, the first Create Table syntax input screen is displayed. You can enter the creator and table names on this screen, as well as the individual column names, formats, and lengths, as shown below:

```
 09:47:19              ***** NATURAL TOOLS FOR DB2 *****              2002-08-12
                              - Create Table -                         1  / 9

>>- CREATE TABLE - SAG_____ . DEMOTABLE_________ ---------------------------->
                    <creator.>table-name
>+--- LIKE ------- ________ . ___________________ +-------------------------+-+>
 !                  <creator.>table/view-name    +- _ - INCLUDING IDENTITY + +
 !                                                                           !
 +( COL1_____________  CHAR___________ ( 20_______ ) _ - __ - _ - __ - _ , +
 +- COL2_____________  INTEGER________ ( _________ ) _ - NN - _ - 2_ - _ , +
 +- COL3_____________  SMALLINT_______ ( _________ ) _ - NN - _ - 1_ - _ , +
 +- COL4_____________  CHAR___________ ( 2________ ) S - __ - _ - __ - _ , +
 +- COL5_____________  VARCHAR________ ( 30_______ ) _ - NN - _ - 3_ - _ , +
 +- COL6_____________  DECIMAL________ ( 2,5______ ) _ - __ - X - __ - _ , +
 +- COL7_____________  FLOAT__________ ( _________ ) _ - NN - _ - __ - _ , +
 +- COL8_____________  DATE___________ ( _________ ) _ - __ - _ - __ - _ , +
 +- COL9_____________  TIME___________ ( _________ ) _ - __ - _ - __ - _ , +
 +- ___________________  _____________ ( _________ ) _ - __ - _ - __ - _ , +
       column-name          format          length    S/M  NN  fld  PK/ R/C
                                                       B       proc UK  D/G
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Exec  Free  --     -     +    ++          Next  Canc
```


Note:
Since the specification of any special characters as part of a Natural field or DDM name does not comply with Natural naming conventions, any special characters allowed within DB2 should be avoided. The same applies to DB2 delimited identifiers, which are not supported by Natural.

In the top right-hand corner of the screen, the index of the top most column (1), and the total number of columns specified (9) is displayed. If you want to specify more columns than fit on one terminal screen, press PF8 to scroll one page forward.

An "S" in the S/M/B field of column 4 means that the FOR SBCS DATA option is selected for this column. Other possible values for this field are M (FOR MIXED DATA) and B (FOR BIT DATA).

Columns 3, 2, and 5 form the primary key, in the specified order. Primary key columns must be selected with an "S" or ordered by specifying appropriate numbers between 1 and 16. In the present example, all primary key columns are defined as "NOT NULL". In addition, column 7 is specified as "NOT NULL".

For column 6, a field procedure has been entered in a window invoked by "S". The window has been closed again, and the "fld proc" field is now marked with "X".

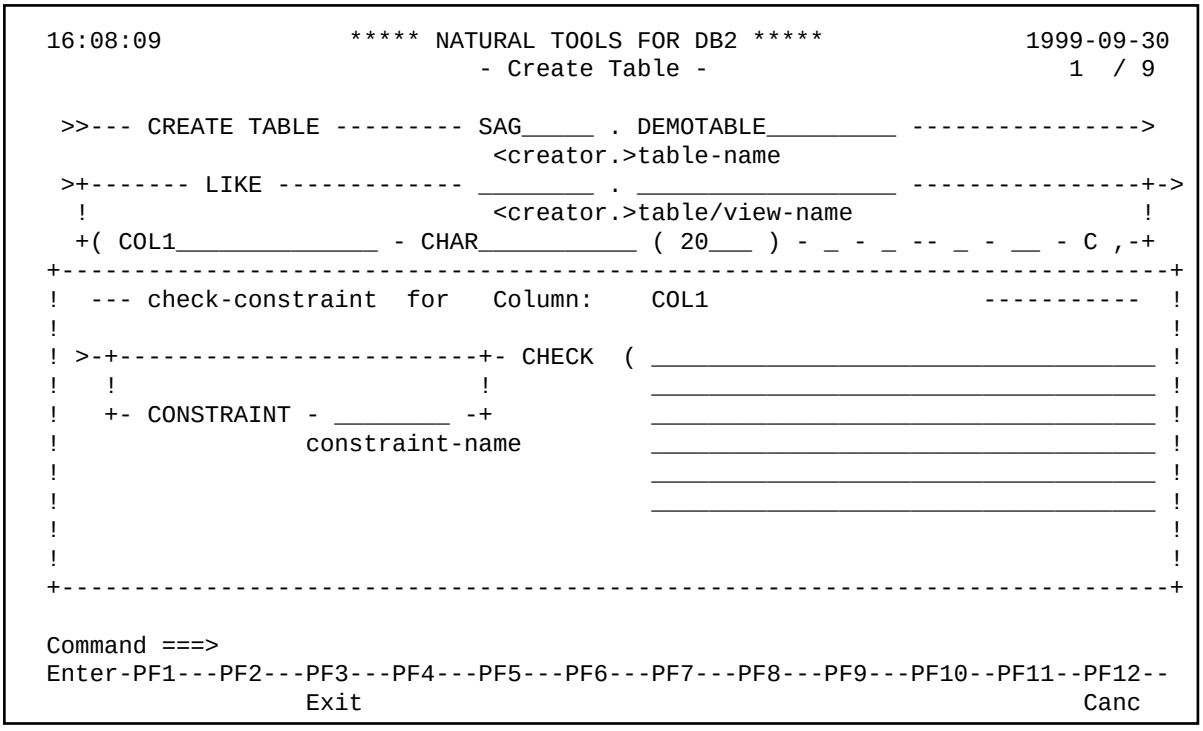
If you enter an "R" in the R/C/D/G field for a given column and press ENTER, a window is displayed, in which you can specify a references clause, which identifies this column as a foreign key of a referential constraint.

```
+-----------------------------------------------------------+
!  References-Clause for Column:  COL1                      !
!                                                           !
! >--- REFERENCES ---- ________ . ____________________ -->  !
!                       <creator.>table-name                !
! >-+---------------------------------+----------------->   !
!   +- ON DELETE --+-- _ - RESTRICT --+                     !
!                  +-- _ - CASCADE ---+                     !
!                  +-- _ - SET NULL --+                     !
!                  +-- _ - NO ACTION -+                     !
!                                                           !
+-----------------------------------------------------------+
```

You must specify the name (with an optional creator name) of the parent table to be referenced. In addition, you must specify the action to be taken when a row in the referenced table is deleted. The following options are provided:

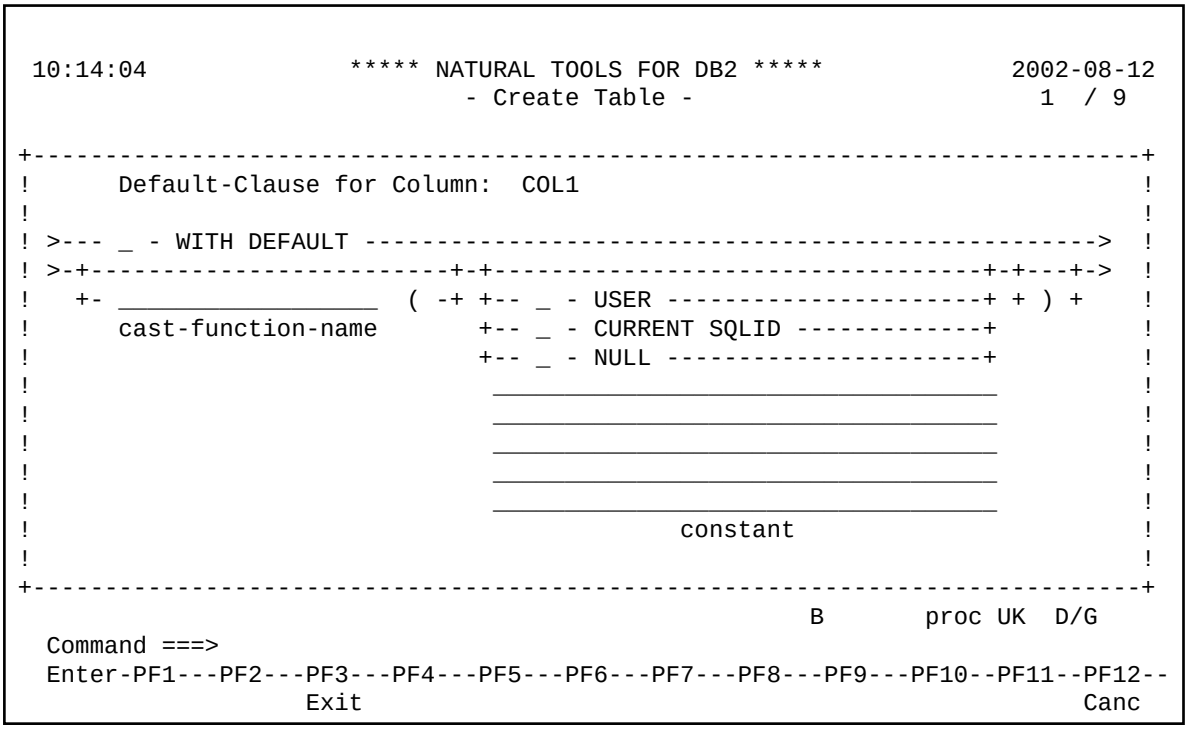
- RESTRICT or NO ACTION prevents the deletion of the parent row until all dependent rows are deleted.
- CASCADE deletes all dependent rows, too.
- SET NULL sets to null all columns of the foreign key in each dependent row that can contain null values.
- A key that consists of more than one column must be defined by a FOREIGN KEY clause.

If you enter a "C" in the R/C/D/G field for a given column and press ENTER, a window is displayed, in which you can specify a check constraint for this column.



You must specify a column check condition. A check condition is a search condition with various restrictions which are described in detail in the relevant DB2 literature by IBM. In addition, you may specify a name for the check constraint.

If you enter a "D" in the R/C/D/G field for a given column and press ENTER, a window is displayed, in which you can specify a default value other than the system default value for this column.

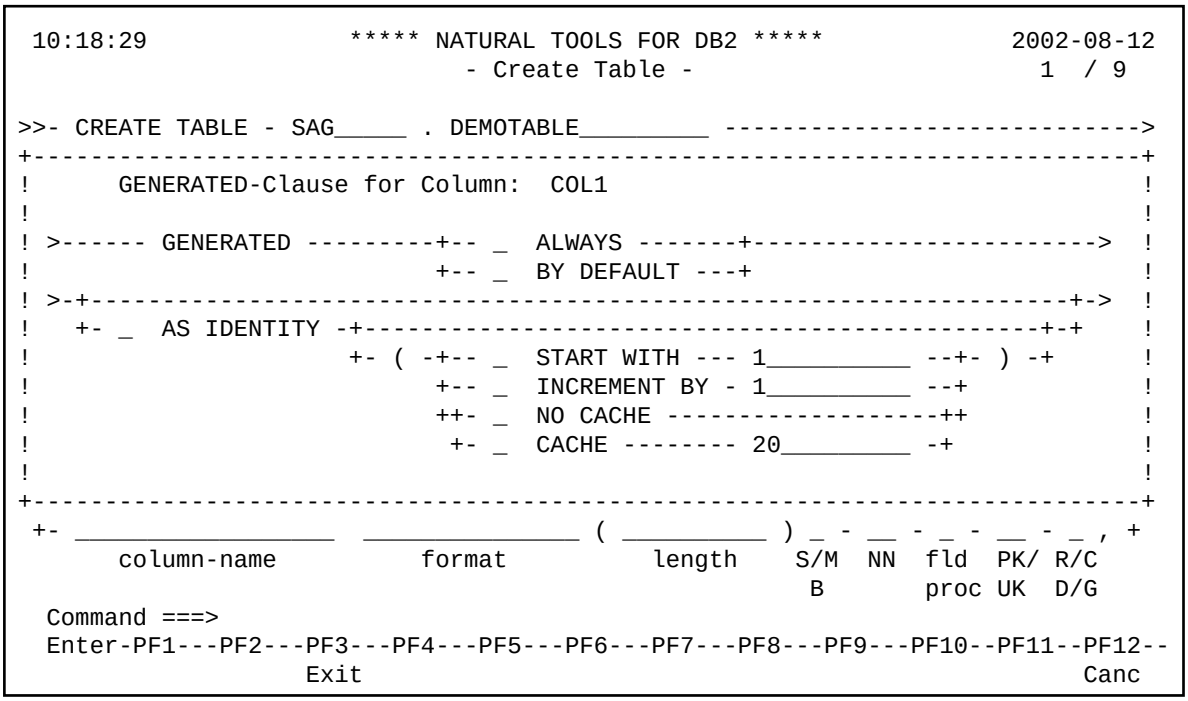


One of the following types of default values can be specified:

- USER: an execution-time value of the special register USER.
- CURRENT SQLID: the SQL authorization ID.
- NULL: the null value.
- constant: a constant which names the default value for the column.

For further information on default values, refer to the relevant DB2 literature by IBM.

If you enter a "G" in the R/C/D/G field for a given column and press ENTER, a window is displayed, in which you can define the GENERATED-Clause for this column.



GENERATED can only be defined if the column has a ROWID data type (or a distinct type that is based on a ROWID data type), or if the column is to be an identity column.

For further information on the GENERATED-Clause, refer to the relevant DB2 literature by IBM.

Windows like the one below may help you in making a valid selection. They are invoked by entering the help character "?" in the appropriate field on the screen:


```
 10:23:44              +----------------------------------+         2002-08-12
                       !  I      INTEGER                  !           1  / 9
                       !  S      SMALLINT                 !
>>- CREATE TABLE - SAG_ !  F      FLOAT(integer)           ! ----------------->
                    <cr !  RE     REAL                     !
>+--- LIKE ------- ____ !  DO     DOUBLE                   ! -------------+-+>
 !                  <cr !  DE     DECIMAL(integer,integer) ! DING IDENTITY + +
 !                      !  N      NUMERIC(integer,integer) !                 !
 +( COL1_____________   !  CH     CHAR(integer)            !  - _ - __ - _ , +
 +- COL2_____________   !  VARC   VARCHAR(integer)         !  - _ - 2_ - _ , +
 +- COL3_____________   !  CL     CLOB(integer)            !  - _ - 1_ - _ , +
 +- COL4_____________   !  B      BLOB(integer)            !  - _ - __ - _ , +
 +- COL5_____________   !  G      GRAPHIC(integer)         !  - _ - 3_ - _ , +
 +- COL6_____________   !  VARG   VARGRAPHIC(integer)      !  - _ - __ - _ , +
 +- COL7_____________   !  DB     DBCLOB(integer)          !  - _ - __ - _ , +
 +- COL8_____________   !  DA     DATE                     !  - _ - __ - _ , +
 +- COL9_____________   !  TIME   TIME                     !  - _ - __ - _ , +
 +- ___________________ !  TIMES  TIMESTAMP                !  - _ - __ - _ , +
       column-name       !                                  !   fld  PK/ R/C
                         !  RO     ROWID                    !   proc UK  D/G
                         !                                  !
Command ===>             !  ______ .. Enter Value          !
Enter-PF1---PF2---PF3-- +----------------------------------+ F10--PF11--PF12---
        Help  Error Exit  Exec  Free  --    -     +     ++          Next  Canc
```

In the case of complex SQL statements, more than one input screen may be required. If so, you can switch to the following screen by pressing PF11 (Next), or return to the previous screen by pressing PF10 (Prev).

As you can see on the above screen, the beginning of the syntax specification for an SQL statement is always indicated by ">>".

Since the syntax of the CREATE TABLE statement is a rather complex one, three more screens are required. Once all necessary information has been entered on the first screen, you press PF11 (Next) to display the next Create Table input screen, where you can specify additional optional parameters.

```
 10:31:51              ***** NATURAL TOOLS FOR DB2 *****              2002-08-12
                                 - Create Table -                       1  / 0

  +-----------------------------------<-----------------------------------+
>-+-+-------------------------------------------------------------------+-+->
    !                                                                     !
    +- , - FOREIGN KEY ----- ________ ------- _ --- (column-name) ->     !
                          <constraint-name>                             !
                                                                        !
     >---- REFERENCES ------ ________ . ____________________ -------->    !
                              <creator.>table-name                      !
                                                                        !
     >-+---------------------------+----- ON DELETE -+- S - RESTRICT -+-+
       +-- _ --- (column-name) ----+                 +- _ - CASCADE --+
                                                     +- _ - SET NULL -+
                                                     +- _ - NO ACTION +




 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec  Free  --    -     +     ++    Prev  Next  Canc
```


On this screen, you can specify a referential constraint to another table. To do so, enter an "S" in the "column-name" field. A list of all columns available in the current table (dependent table) is displayed, where you can select the column(s) to comprise the foreign key related to another table (parent table). You can also specify a name for the constraint. If not, the constraint name is derived from the first column of the foreign key.

A foreign key consists of one or more columns in a dependent table that together must take on a value that exists in the primary key of the related parent table.

In the REFERENCES part, you must specify the table name (with an optional creator name) of the parent table which is to be affected by the specified constraint. In addition, you must specify the action to be taken when a row in the referenced parent table is deleted.

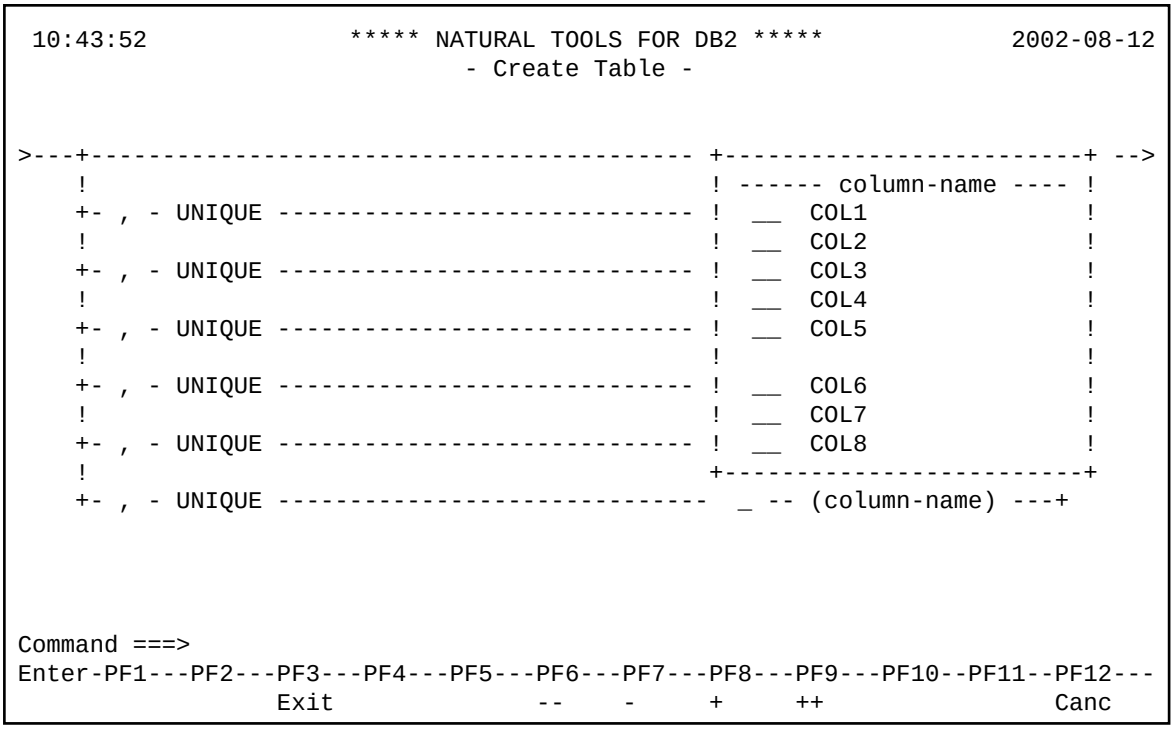
The following options are provided:

- RESTRICT or NO ACTION prevents the deletion of the parent row until all dependent rows are deleted.
- CASCADE causes all dependent rows to be deleted, too.
- SET NULL sets to null all columns of the foreign key in each dependent row that can contain null values.

In the top right-hand corner of the screen, the index of the currently displayed referential constraint block (1) and the total number of referential constraint blocks defined (0) is displayed.

When all information has been entered, you can press either PF10 (Prev) to return to the previous screen, or PF11 (Next) to go to the next screen.

On the next screen you have again the possibility to specify columns as unique. This time, however, up to six groups of unique columns can be defined, with up to 16 columns per group. The individual columns are specified in a window, which can be invoked for each group.



Since unique columns must not contain null values, a further window is invoked automatically, on which you can define the columns specified as unique also as NOT NULL (unless you already defined them as such on the first Create Table input screen).

When all information has been entered, you can press either PF10 (Prev) to return to the previous screen or PF11 (Next) to go to the last syntax input screen as shown below:

```
10:47:02              ***** NATURAL TOOLS FOR DB2 *****             2002-08-12
                            - Create Table -

  >-----+- IN -----------  _________ . _________ ----------------------+------>
        !             <database-name.>tablespace-name                  !
        +- IN DATABASE -------------- _________ ----------------------+
                                    database-name

  >------- EDITPROC ------ _________ --------- VALIDPROC -- _________ -------->

  >------- AUDIT --------- ________ ---------- OBJID ------ _____ ------------>
                   ( NONE, CHANGES, ALL )                  integer

  >------- DATA CAPTURE -- ________ ---------- CCSID ------ ________ --------->
                       ( NONE, CHANGES )               ( ASCII, EBCDIC )

  >------- WITH RESTRICT ON DROP -- _ ------------------------------------->

  >------- CHECK --------- _ ------------------------------------------------><
                    check-condition

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
       Help  Error Exit  Exec  Free                           Prev        Canc
```

On this screen, you can now:

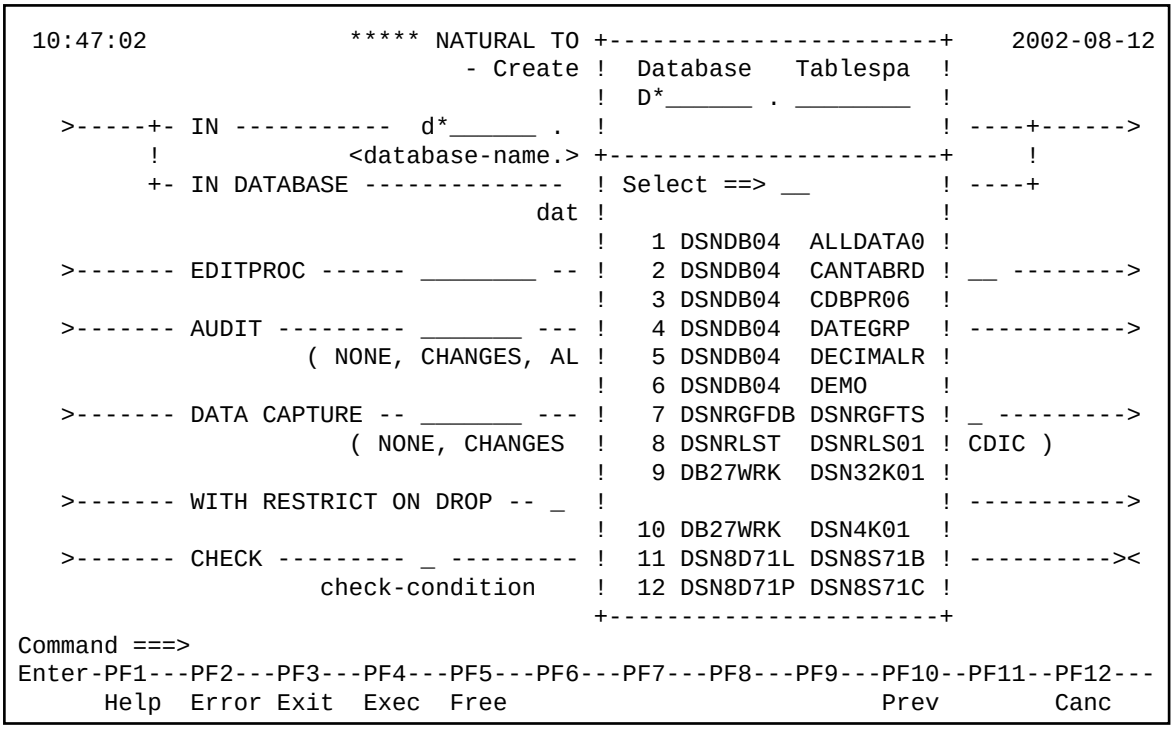
- Restrict dropping of the current table (and also of the database and tablespace that contain this table).
- Define a check constraint for the current table.
  To define a check constraint, you must specify a table check condition. A check condition is a search condition with various restrictions which are described in the relevant DB2 literature by IBM. In addition, you may specify a name for the check constraint.

If you press PF10 (Prev) on this screen, you return to the previous screen.

As you can see on the above screen, the end of the syntax specification for an SQL statement is always indicated by "><".

An active help facility that consists of selection lists in windows is available for all fields referencing existing database objects. Selection lists are invoked by entering either an asterisk (*) or part of an object name followed by an asterisk in the corresponding input field.

If, for example, you enter "D*" in the "database-name" field of the above screen, a window appears where you can check your selection criteria. When you press ENTER, a list of all databases whose names begin with "D" appears.



Within the selection list, you can scroll up (PF6 / "--" or PF7 / "-") or down (PF8 / "+" or PF9 / "++"), and select the desired database. The name of the selected database is copied to the corresponding field in your input screen.

When all information has been entered, you can either switch to free mode (PF5) or submit the created member directly to DB2 for execution (PF4). If execution is successful, you receive the message:

Statement(s) successful, SQLCODE = 0

If not, an error code is returned.

In free mode, the following editor screen displays the generated SQL code:


```
 10:53:50              ***** NATURAL TOOLS FOR DB2 *****             2002-08-12
 FREE - Input        SAG                   S 01- ---------------Columns 001 072
=====>                                                          Scroll ===>  PAGE
 ***** ***************************** top of data ****************************
 00001 CREATE TABLE SAG.DEMOTABLE
 00002   (COL1                 CHAR(20),
 00003    COL2                 INTEGER                NOT NULL,
 00004    COL3                 SMALLINT               NOT NULL,
 00005    COL4                 CHAR(2)                FOR SBCS DATA,
 00006    COL5                 VARCHAR(30)            NOT NULL,
 00007    COL6                 DECIMAL(2,5)
 00008      FIELDPROC PROGNAME
 00009        ('STRING1','STRING2'),
 00010    COL7                 FLOAT                  NOT NULL,
 00011    COL8                 DATE,
 00012    COL9                 TIME,
 00013   PRIMARY KEY (COL3,                    COL2,
 00014                COL5)
 00015   )
 00016   IN DSNDB04.DEMO;
 ***** ************************** bottom of data ****************************

  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Setup Exit  Exec  Rfind Rchan -     +     Outpu             Canc
```

The free-mode editor is an adapted version of the Software AG Editor. It is almost identical to the interactive SQL input screen. However, no SELECT statements can be issued from free mode.

For further details, please refer to the relevant Software AG Editor documentation.

Create Tablespace Function

If you enter the object code "TS" in the CREATE function, the first Create Tablespace syntax input screen is displayed:

```
 16:08:09              ***** NATURAL TOOLS FOR DB2 *****              1999-08-15
                              - Create Tablespace -

  >>-- CREATE TABLESPACE ----- TS1_____ -------- IN ----- _________ ----------->
                           tablespace-name             database-name

            +- VCAT ---- _________ ------------------------------------------+
  >- USING -+          catalog-name                                          +->
            +- STOGROUP- _________ - PRIQTY  ____ - SECQTY  ___ - ERASE  ___ -+
                         stogroup-name      integer       integer   ( YES or NO )


  >--- FREEPAGE -------- ___ ----- PCTFREE -- __ ------------ COMPRESS  ___ --->
                         integer            integer                ( YES or NO )

  >--- NUMPARTS -------- __ ----- _ ------------------------------------------>
                         integer    PART

  >--- SEGSIZE --------- __ ------------------------------------------------->
                         integer

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
       Help  Error Exit  Exec  Free                                 Next  Canc
```


Once you have entered all necessary information, press PF11 (Next) to go to the next screen:

```
16:08:09              ***** NATURAL TOOLS FOR DB2 *****             1999-08-15
                             - Create Tablespace -


>---+-------------------------------------------------------------------+--><
    !                                                                   !
    +--- BUFFERPOOL ------ ______ ------------------------------------- +
    !                     bufferpool-name                               !
    +--- LOCKSIZE -+------ _________ ------------------------+----------+
    !              !( ANY, TABLE, TABLESPACE )               !          !
    !              +------ ____ ---+-----------------------+--+          !
    !               ( ROW or PAGE )!                       !             !
    !                              +- LOCKMAX -- ______ --+              !
    !                                     ( SYSTEM or integer )          !
    +--- CLOSE ----------- ___ ------------------------------------------+
    !                ( YES or NO )                                      !
    +--- DSETPASS -------- ________ ------------------------------------+
                           password



Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
      Help  Error Exit  Exec  Free                          Prev        Canc
```

On the second Create Tablespace syntax input screen, you can now specify additional buffer pool names as well as the LOCKSIZE option with the LOCKMAX clause.

If you enter an "S" in the "bufferpool-name" field and press ENTER, a window is displayed, in which you can specify additional buffer pool names.

Refer to the relevant DB2 literature by IBM for further details on the COMPRESS, LOCKSIZE and LOCKMAX clauses.

Alter Table Function

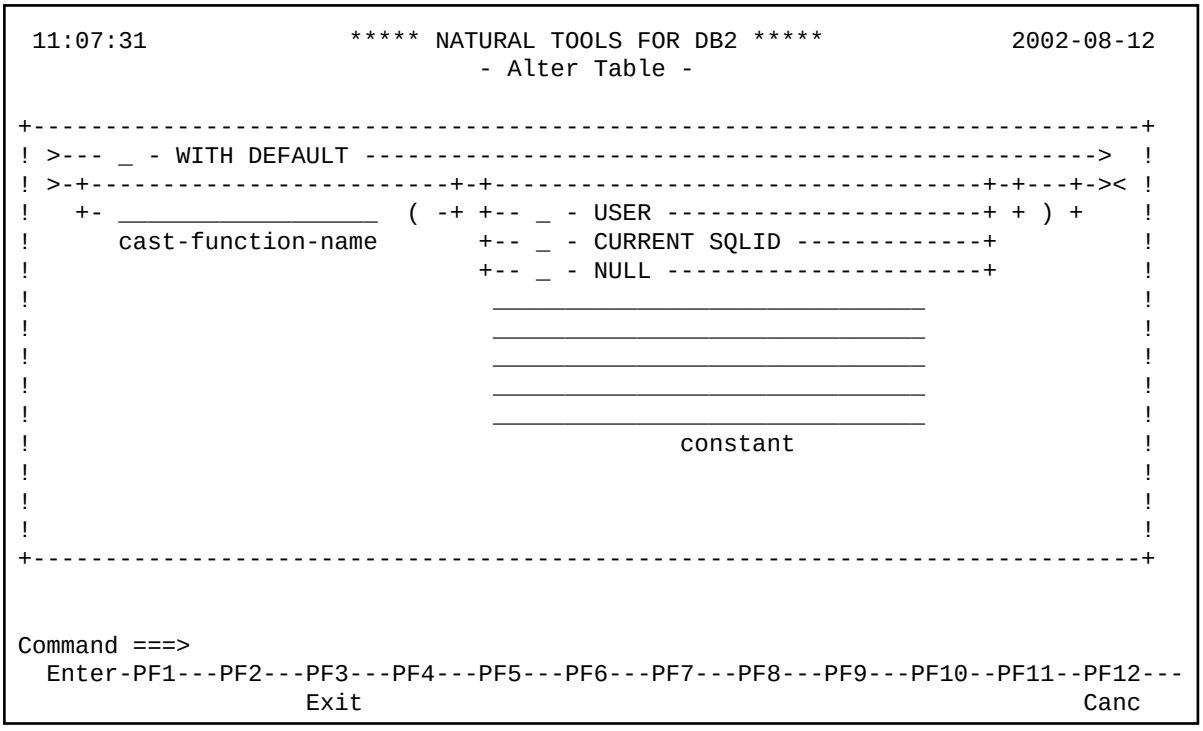
If you invoke the Alter Table syntax input screen, you can specify the following:


```
11:01:47              ***** NATURAL TOOLS FOR DB2 *****             2002-08-12
                                - Alter Table -

>>-- ALTER TABLE ------------ ________ . ____________________ ----------------->
                              <creator.>table-name
>-+- ALTER  ___________________ -- SET DATA TYPE - VARCHAR - ( __________ ) --+>
  !             column-name                                       length       !
>-+- ADD  ____________________ _________________  ( __________ ) - _ -- __ - __ -->
  !          column-name          format            length     S/M/B  NN  UK/PK
  !    +--<
  !    +>- _ ------ _ ---------- _ -------------- _ -------------- _ -------+>
  !   field-proc default   check-constr   reference-constr   GENERATED-Clause!
  !                                                                          !
>-+- VALIDPROC --------------- _________ ------------------------------------+>
  !                         program-name or NULL                             !
  +- AUDIT ------------------- ________ ------------------------------------+
  !                         ( NONE, CHANGES, ALL )                           !
  +- DATA CAPTURE ------------ ________ ------------------------------------+
                             ( NONE, CHANGES )


  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec  Free                                Next  Canc
```

If you enter an "S" in the "field-proc" input field and press ENTER, a window is displayed, in which you can specify a field procedure to be executed for this column:

```
 11:05:47              ***** NATURAL TOOLS FOR DB2 *****             2002-08-12
                                - Alter Table -

>>-- ALTER TABLE ------------ ________ . ____________________ ----------------->
                              <creator.>table-name
>-+- ALTER  ___________________ -- SET DATA TYPE - VARCHAR - ( __________ ) --+>
  !             column-name                                       length       !
>-+- ADD  ____________________ _________________  ( __________ ) - _ -- __ - __ -->
  !          column-name          format            length     S/M/B  NN  UK/PK
  !    +--<              +-----------------------+
  !    +>- S ------ _ -- !             1 / 0     !  --------------- _ -------+>
  !   field-proc default ! --- FIELDPROC ----    ! -constr   GENERATED-Clause!
  !                      !      _________         !                          !
>-+- VALIDPROC --------- !    program-name        ! --------------------------+>
  !                      !  ( _____________ ,     !                          !
  +- AUDIT -------------- !    _____________ ,     ! --------------------------+
  !                      !    _____________ )     !                          !
  +- DATA CAPTURE ------ !    (constants,)        ! --------------------------+
                         !                        !
                         !                        !
                         +------------------------+
Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                    Exit                                                  Canc
```

If you enter an "S" in the "default" field and press ENTER, a window is displayed, in which you can specify a default value other than the system default value for this column:

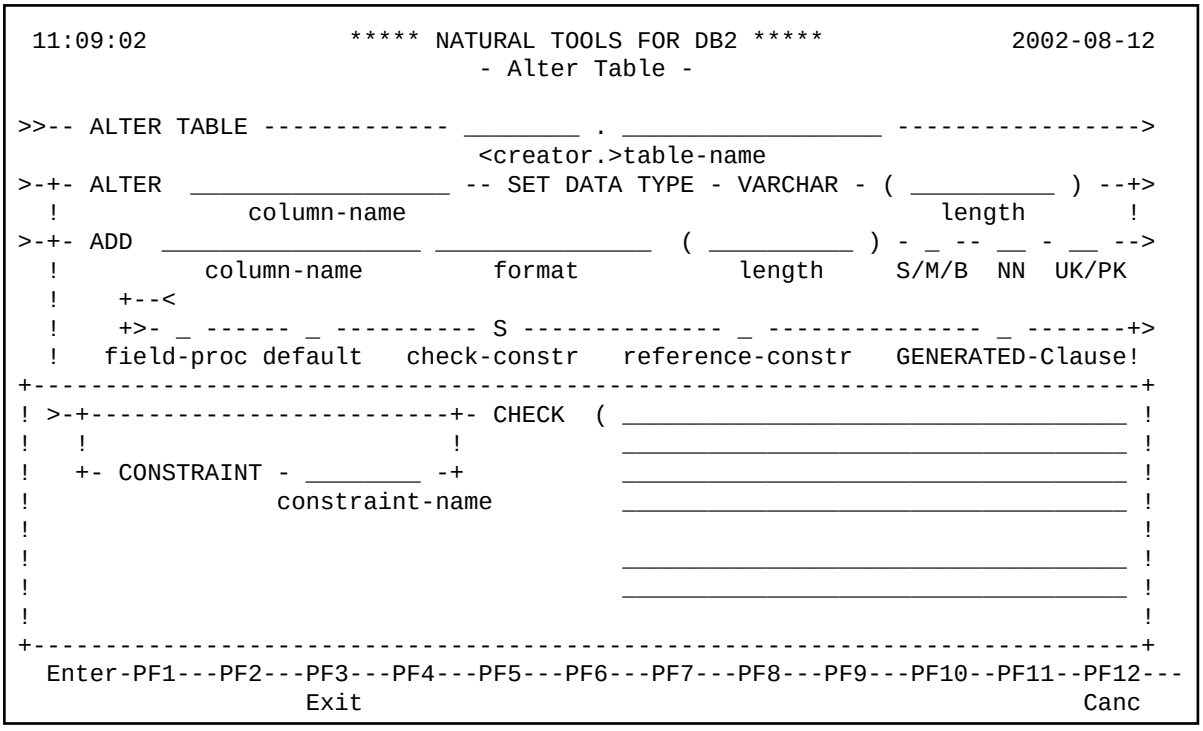


One of the following types of default values can be specified:

- USER: an execution-time value of the special register USER.
- CURRENT SQLID: the SQL authorization ID.
- NULL: the null value.
- constant: a constant which names the default value for the column.

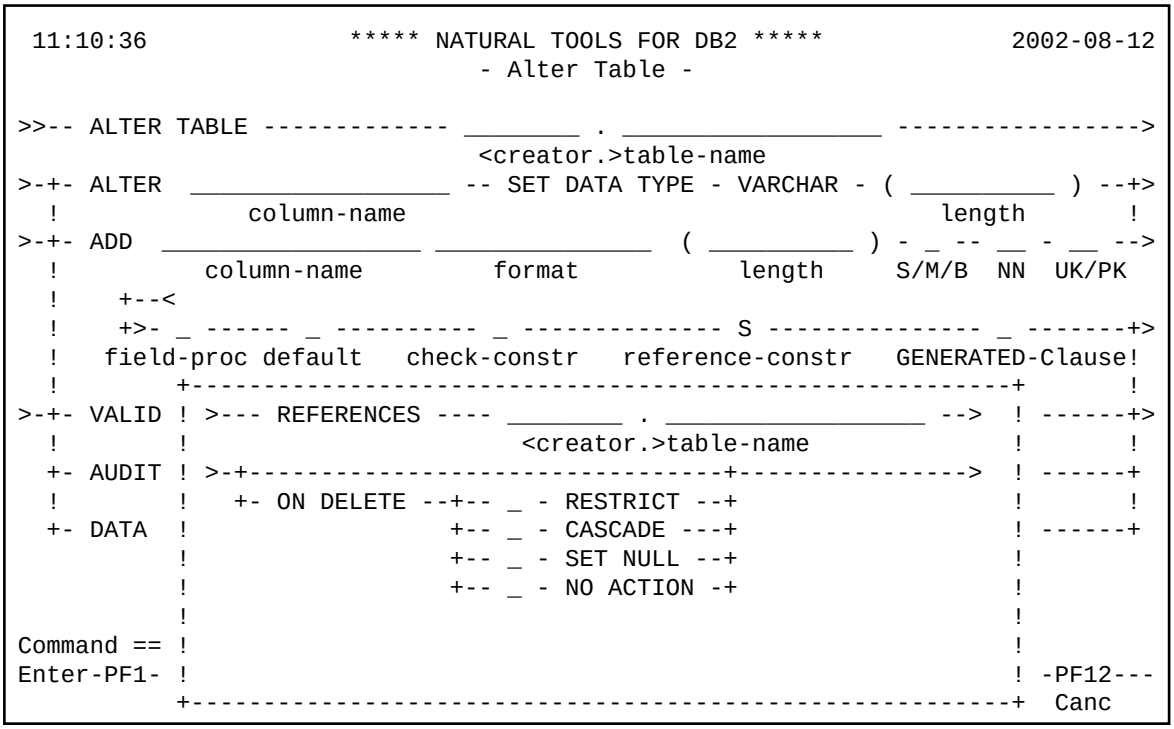
For further information on default values, refer to the relevant DB2 literature by IBM.

If you enter an "S" in the "check-constraint" field and press ENTER, a window is displayed, in which you can specify a check constraint for this column:



You must specify a column check condition. A check condition is a search condition with various restrictions which are described in detail in the relevant DB2 literature by IBM. In addition, you may specify a name for the check constraint.

If you enter an "S" in the "reference-constraint" field and press ENTER, a window is displayed, in which you can specify a references clause, which identifies this column as a foreign key of a referential constraint:



You must specify the name (with an optional creator name) of the parent table to be referenced. In addition, you must specify the action to be taken when a row in the referenced table is deleted. The following options are provided:

- RESTRICT or NO ACTION prevents the deletion of the parent row until all dependent rows are deleted.
- CASCADE deletes all dependent rows, too.
- SET NULL sets to null all columns of the foreign key in each dependent row that can contain null values.

Once you have entered your column definitions, press PF11 (NEXT).

A screen is invoked in which you can add or drop primary and/or foreign keys:

```
 11:14:42              ***** NATURAL TOOLS FOR DB2 *****             2002-08-12
                                - Alter Table -

>--+--- ADD ------ PRIMARY KEY ------------------- _ -- (column-name) ---+
   !                                                                      !
   +--- DROP ----- PRIMARY KEY ------------------- _ --------------------+-->


>--+->- ADD ------ FOREIGN KEY --- ________ ------ _ -- (column-name) -->
   !                              constraint-name
   ! >- REFERENCES ----> ________ . ___________________ ----------------->
   !                       <creator.>table-name
   ! >-+---------------------------+------ ON DELETE -+- S - RESTRICT -+-+-->
   !   +--- _ --- (column-name) ---+                  +- _ - CASCADE --+ !
   !                                                  +- _ - SET NULL -+ !
   !                                                  +- _ - NO ACTION + !
   +->- DROP ----- FOREIGN KEY --- ________ ----------------------------+
                                  constraint-name



  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec  Free                          Prev  Next  Canc
```

Once you have entered the required information for adding and/or dropping primary and/or foreign keys, press PF11 (NEXT). A screen is invoked, in which you can specify a RESTRICT ON DROP clause, add or drop a CHECK constraint, and/or drop any constraint:


```
12:20:24              ***** NATURAL TOOLS FOR DB2 *****             1999-08-15
                                - Alter Table -


>---+-- ADD --- _ --+----- RESTRICT ON DROP ---------------------------------->
    !               !
    +-- DROP -- _ --+


>------ ADD CHECK ----------- _ ------------------------------------------------>
                             check-condition

>------ DROP CHECK  --------- _________ ------------------------------------->
                             constraint-name


>------ DROP CONSTRAINT ----- _________ ------------------------------------->
                             constraint-name



Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Exec  Free                           Prev        Canc
```

Alter Tablespace Function

If you invoke the Alter Tablespace syntax input screen, you can specify the following:

```
12:20:24              ***** NATURAL TOOLS FOR DB2 *****              1999-08-15
                               - Alter Tablespace -

 >>-------------- ALTER TABLESPACE -- _________ . _________ -------------------->
                                    <database-name.>tablespace-name

            +-->- BUFFERPOOL ------- _______ ----------------+
            !                       bufferpool-name           !
  >---------+-->- CLOSE ------------ ___ --------------------+--------------->
            !                       ( YES or NO )             !
            +-->- DSETPASS --------- ________ ---------------+
            !                          password               !
            +-->- PART ------------- __ ---------------------+
            !                          integer                !
            +-->- FREEPAGE --------- ___ --------------------+
            !                          integer                !
            +-->- PCTFREE ---------- __ ---------------------+
            !                          integer                !
            +-->- COMPRESS --------- ___ --------------------+
                                    ( YES or NO )

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Exec  Free                                 Next  Canc
```

If you enter an "S" in the "bufferpool-name" field and press ENTER, a window is displayed, in which you can specify additional buffer pool names:


```
12:20:24             ***** NATURAL TOOLS FOR DB2 *****              2002-08-28
                            - Alter Tablespace -
                                                 +--------------------------------+
 >>------------- ALTER TABLESPACE -- _______ !                                    !
                               <database-na !     Valid values for              !
                                            !     bufferpool-name:              !
            +-->- BUFFERPOOL ------- S_____  ! ----------------------------     !
            !                  bufferpool-n !                                    !
  >---------+-->- CLOSE ------------ ___ --- !  -  4KB buffer pools -           !
            !                  ( YES or NO  !  BP0, BP1, BP2, ..., BP49         !
            +-->- DSETPASS --------- _______ !                                   !
            !                        passwor !  - 32KB buffer pools -            !
            +-->- PART ------------- __ ---- !  BP32K, BP32K1, ..., BP32K9      !
            !                      integer   !                                   !
            +-->- FREEPAGE --------- ___ --- !  ______  Selection               !
            !                      integer   !                                   !
            +-->- PCTFREE ---------- __ -----+-------------------------------+
            !                      integer                   !
            +-->- COMPRESS --------- ___ --------------------+
                               ( YES or NO )

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit                                                  Canc
```

Once you are back in the first Alter Tablespace syntax input screen, press PF11 (Next) to go to the next screen:

```
12:20:24             ***** NATURAL TOOLS FOR DB2 *****              1999-08-15
                               - Alter Tablespace -


                         +- VCAT ----- _________ --+
            +-->- USING -+          catalog-name  +---------------+
            !            +- STOGROUP - _________ --+               !
            !                       stogroup-name                  !
  >---------+-->- PRIQTY ------------ ____ ----------------------+------><
            !                        integer                      !
            +-->- SECQTY ------------ ___ ----------------------+
            !                        integer                      !
            +-->- ERASE ------------- ___ ----------------------+
            !                      (YES or NO)                    !
            +-->- LOCKMAX ----------- ______ --------------------+
            !                      (SYSTEM or integer)            !
            +-->- LOCKSIZE ---+------- ____ --- LOCKMAX - ______ -+
                              ! (PAGE or ROW)  (SYSTEM or integer)!
                              +------- __________ ---------------+
                                   (ANY, TABLE or TABLESPACE)

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Exec  Free                          Prev        Canc
```

On the second Alter Tablespace syntax input screen, you can now specify the LOCKMAX and LOCKSIZE options.

Refer to the relevant DB2 literature by IBM for further details on the COMPRESS, LOCKSIZE and LOCKMAX clauses.

SQL Skeleton Members

SQL Skeleton Members

SQL skeleton members are provided for processing the following SQL statements that are not supported by the Catalog Maintenance function:

CREATE AUXILIARY TABLE, CREATE DISTINCT TYPE, CREATE TRIGGER, GRANT ALTERIN, REVOKE ALTERIN.

An SQL skeleton member is a Natural Text member that contains an SQL skeleton that complies with the DB2 SQL syntax rules as described in the relevant IBM literature. The replacable items in the SQL skeleton shown in lower-case characters must be filled with user input so that the skeleton becomes a valid SQL statement that can be executed in Free Mode (as described above) or ISQL (see Interactive SQL). The skeleton members are delivered in the Natural system library SYSDB2, along with example SQL members.

NDB - Procedure Maintenance

As described below, this function does not apply to all DB2 versions:

For DB2 for OS/390 Version 5 and earlier versions, the Procedure Maintenance function of the Natural Tools for DB2 can be used to insert, modify and delete stored procedures from the SYSIBM.SYSPROCEDURES table, as well as insert data areas.

From DB2 UDB for OS/390 Version 6 onwards, the handling of stored procedures in DB2 has changed, and stored procedures can no longer be maintained with the Procedure Maintenance function. Instead, for maintaining stored procedures, DB2 provides the new statements CREATE PROCEDURE and ALTER PROCEDURE as described in the relevant DB2 literature by IBM.

This section covers the following topics:

- Invoking Procedure Maintenance
- Insert a Stored Procedure
- Modify a Stored Procedure
- Insert Data Areas
- Save PARMLIST as Natural Object
- List Stored Procedures
- Delete a Stored Procedure

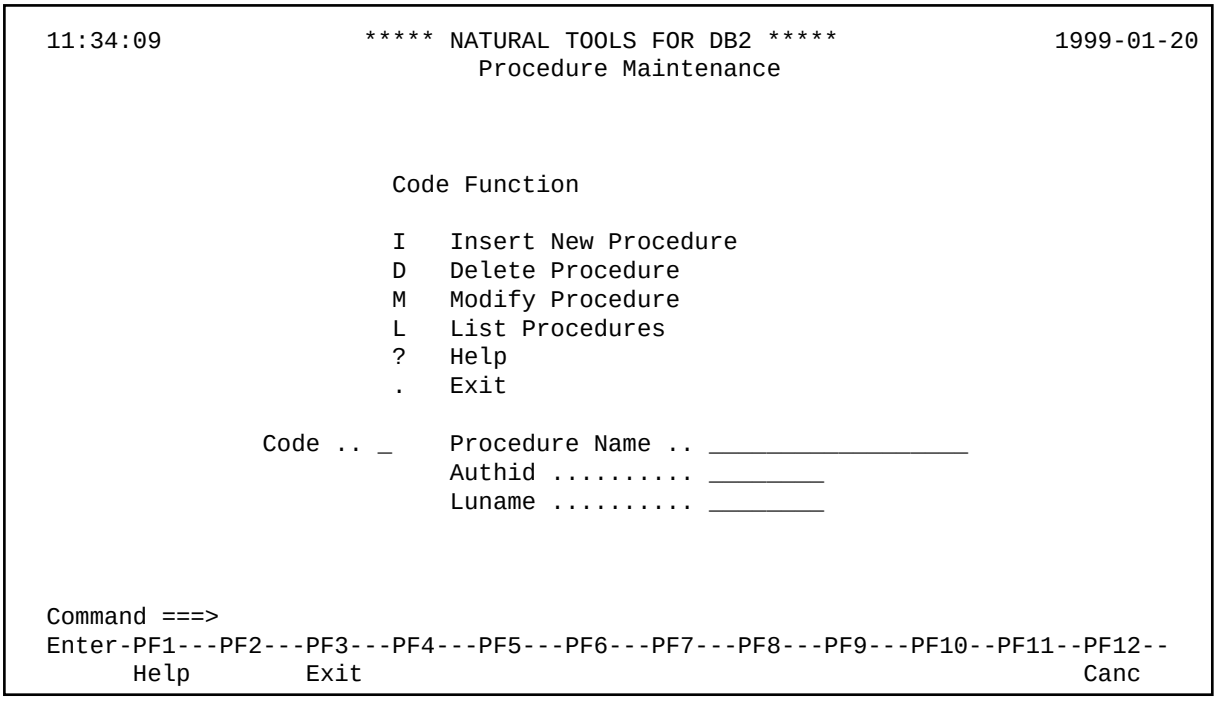
See also Natural Stored Procedures in the section Statements and System Variables.

Invoking Procedure Maintenance

To invoke the Procedure Maintenance function

- Enter function code "P" on the Natural Tools for DB2 Main Menu.

The Procedure Maintenance menu is displayed:



The following functions are available:

Code	Description
I	Inserts a new DB2 stored procedure into the SYSIBM.SYSPROCEDURES table.
D	Deletes one or several DB2 stored procedure(s).
M	Modifies a DB2 stored procedure.
L	Lists all DB2 stored procedures or a part of them.

The following parameters can be specified:

Parameter	Description
Procedure Name	Specifies the name of the procedure.
Authid	Specifies the authorization ID.
Luname	Specifies the LU name.

Insert a Stored Procedure

To insert a new stored procedure into the catalog

- Enter function code "I" on the Procedure Maintenance menu.

The following screen is displayed:

```
14:53:28              ***** NATURAL TOOLS FOR DB2 *****              1999-08-04
                             - Insert Procedure -

  PROCEDURE = DBA _______________          LOADMOD = ________
  AUTHID    = _________                    LINKAGE = _ (N,blank)
  LUNAME    = _________                     COLLID = ___________________
  ASUTIME   = __________              STAYRESIDENT = _ (Y,blank)
  IBMREQD   = _ (Y/N)                  RESULT_SETS = _____
  PGM_TYPE  = _ (M/S)            EXTERNAL_SECURITY = _ (Y/N)
  WLM_ENV   = ___________________ COMMIT_ON_RETURN = _ (Y/N)
  LANGUAGE  = _________ (ASSEMBLER,PLI,COBOL,C)

  RUNOPTS   = ' ____________________________________________________
              ____________________________________________________
              ____________________________________________________
              ____________________________________________________
              ____________________________________________________ ,
  PARMLIST: -   define PARMLIST: N (Y/N)

Command ===>

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Exec                                          Canc
```

Here, you can specify the fields of the new stored procedure. The fields PROCEDURE, LOADMOD, IBMREQD and LANGUAGE are mandatory.

To define a parameter list

- Enter "y" in the field "Define PARMLIST".

You can then enter the parameter fields in a separate screen.

The first column in the parameter screen is used for editing purposes, possible commands are "I" and "D" for insertion and deletion. In the second column, the name of the parameter can be entered. The third column is mandatory and specifies the type of the parameter. Entering "?" displays a list of possible types from which one type may be selected. In the next column, a length can be entered. The Natural for DB2 checks whether the specified type and length value fit together. The next columns define the subtype and the in/out value.

When all parameters are specified, press ENTER to return to the Insert Procedure screen.

To actually insert the stored procedure and add it to the catalog, press PF4. If pressing PF4 results in an error from DB2, press PF2 to display the error information.

Modify a Stored Procedure

To modify a stored procedure in the catalog

- Enter function code "M" in the Procedure Maintenance menu together with a procedure name as unique identifier.

The following screen is displayed:

```
14:55:57              ***** NATURAL TOOLS FOR DB2 *****             1999-08-04
                                - Modify Procedure -

  PROCEDURE = DBAPROC___________          LOADMOD = DBAMOD__
  AUTHID    = _________                  LINKAGE = _ (N,blank)
  LUNAME    = _________                   COLLID = ____________________
  ASUTIME   = __________              STAYRESIDENT = _ (Y,blank)
  IBMREQD   = Y (Y/N)                  RESULT_SETS = _____
  PGM_TYPE  = _ (M/S)             EXTERNAL_SECURITY = _ (Y/N)
  WLM_ENV   = ____________________ COMMIT_ON_RETURN = _ (Y/N)
  LANGUAGE  = C_______ (ASSEMBLER,PLI,COBOL,C)

  RUNOPTS   = ' ____________________________________________________
                ____________________________________________________
                ____________________________________________________
                ____________________________________________________
                ____________________________________________________ ,
  PARMLIST: X   display PARMLIST: N (Y/N)

 Command ===>

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec                                          Canc
```

To display a list of stored procedures

- Proceed as above but do not enter a procedure name nor use the asterisks"*" notation.

The list of stored procedures will be displayed.

From this list select a specific stored procedure with the line command "MO".

The stored procedure is then displayed for modification. Proceed as described in section Insert a Stored Procedure above.

Insert Data Areas

Fields of data areas can be inserted automatically to SYSPROCEDURES.

Proceed as described in the section Insert a Stored Procedure or Modify a Stored Procedure**.**

```
 14:57:16              ***** NATURAL TOOLS FOR DB2 *****            1999-08-04
  +-----------------------------------------------------------------+
  !  PARMLIST =                                                     !
  !                                                                 !
  !   _  ________  _______________ ( _____ ) FOR  _____  DATA  _____  !
  !   _  ________  _______________ ( _____ ) FOR  _____  DATA  _____  !
  !   _  ________  _______________ ( _____ ) FOR  _____  DATA  _____  !
  !   _  ________  _______________ ( _____ ) FOR  _____  DATA  _____  !
  !   _  ________  _______________ ( _____ ) FOR  _____  DATA  _____  !
  !   _  ________  _______________ ( _____ ) FOR  _____  DATA  _____  !
  !   _  ________  _______________ ( _____ ) FOR  _____  DATA  _____  !
  !   _  ________  _______________ ( _____ ) FOR  _____  DATA  _____  !
  !   _  ________  _______________ ( _____ ) FOR  _____  DATA  _____  !
  !   _  ________  _______________ ( _____ ) FOR  _____  DATA  _____  !
  !  -------------------------------------------------------------- !
  !  cmd parm-name parm-type        length       subtype       inout  !
  !                                                                 !
  !   I/ optional  mandatory                     blank/        blank/ !
  !   D                                          SBCS/         IN/    !
  !   I=insert                                   MIXED/        OUT/   !
  +-----------------------------------------------------------------+
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit  Parml Data        -     +                       Canc
```

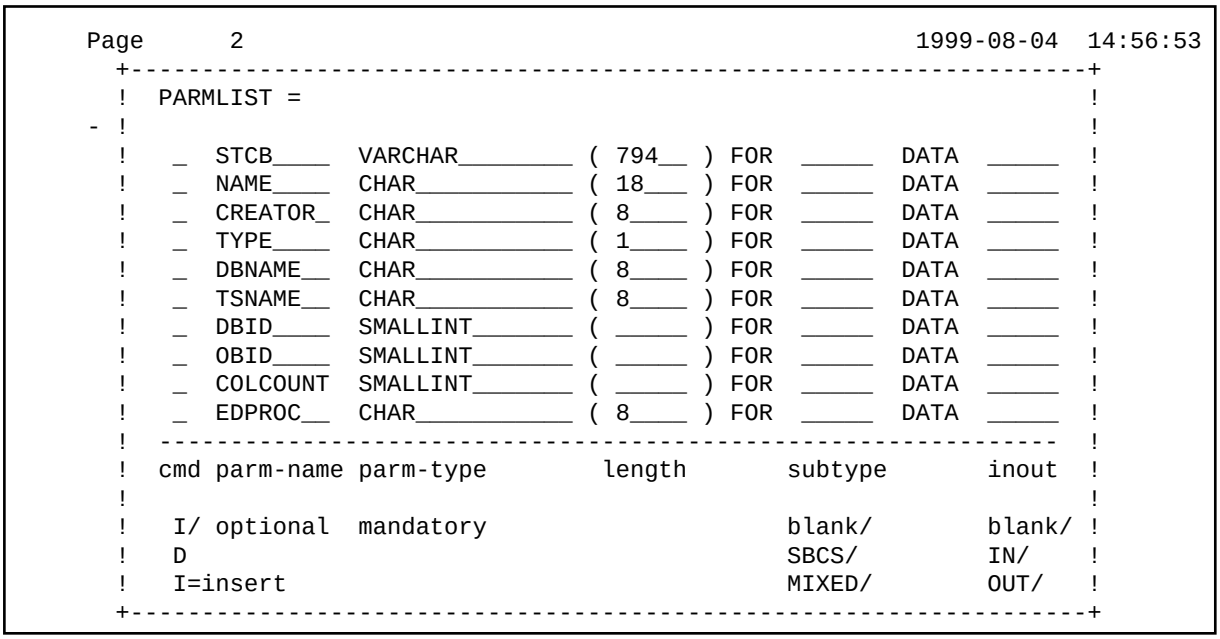
In the PARMLIST screen (above,) press PF5/Data to display the following window.

```
 14:57:16              ***** NATURAL TOOLS FOR DB2 *****            1999-08-04
  +-----------------------------------------------------------------+
  !  PARMLIST =                                                     !
  !                                                                 !
  !   _  ________  _______________ ( _____ ) FOR  _____  DATA  _____  !
  !   _  ________  _______________ ( _____ ) FOR  _____  DATA  _____  !
  !   _  ________  __________ +--------------------------------+ ___  !
  !   _  ________  __________ !                                ! ___  !
  !   _  ________  __________ ! Library :  ________            ! ___  !
  !   _  ________  __________ ! Member  :  ________            ! ___  !
  !   _  ________  __________ !                                ! ___  !
  !   _  ________  __________ !                                ! ___  !
  !   _  ________  __________ ! Enter Library and LDA/PDA name ! ___  !
  !   _  ________  __________ ! and press enter to insert      ! ___  !
  !  ----------------------- ! press PF3 to cancel            ! ---  !
  !  cmd parm-name parm-type  +--------------------------------+ out  !
  !                                                                 !
  !   I/ optional  mandatory                     blank/        blank/ !
  !   D                                          SBCS/         IN/    !
  !   I=insert                                   MIXED/        OUT/   !
  +-----------------------------------------------------------------+
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit  Parml  Data        -     +                       Canc
```


Enter the library name and the name of the LDA or PDA to be inserted into the PARMLIST definition of the stored procedure. A maximum of 100 parameters can be inserted.

The parameter STCB will be generated automatically at the top of the PARMLIST.

The individual field names of the LDA or PDA are truncated to the first 8 characters in the PARMLIST parmnames (DB2 restriction).



Save PARMLIST as Natural Object

You can store the field definitions of the PARMLIST in a Natural program which you can stow and modify. (It is not necessary to enter the field definitions manually.)

To do so, press PF4/Parml.

The field definitions are converted to a Natural object. The Natural object is saved on FUSER in the user library you specified.

The Natural object can be used as input for the LDA editor after the stow.

List Stored Procedures

To display a list of stored procedures

- Enter function code "L" in the Procedure Maintenance menu.

The following screen is displayed:

```
11:17:05              ***** NATURAL TOOLS FOR DB2 *****             1999-01-21
  PROCEDURE PROC*.SAG.*                      S 01      Row 0 of 3 Columns 041075
  ====>                                                Scroll ===> PAGE
             PROCEDURE   AUTHID   LUNAME     LOADMOD L           COLLID  LANGU
  ** ****************************** top of data*******************************
  __               PROC1      SAG              MODULE1                      ASSEM
  __               PROC2      SAG              MODULE2
  __               PROC3      SAG              MODULE3
  ** ***************************** bottom of data*****************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +           <     >    Canc
```

The screen contains a list of all stored procedures in the catalog.

To display a part of the procedures

- use the asterisks "*" notation.

The following line commands are available:

Command	Description
LI	Display the stored procedure in detail
MO	Modify the stored procedure
DE	Delete the stored procedure

Delete a Stored Procedure

To delete a stored procedure from the catalog

- Enter function code "D" in the Procedure Maintenance menu together with a procedure name as unique identifier.

The procedure is then deleted.

If you do not enter a procedure name or if you use the "*" notation, a screen with a list of procedures appears, where you can enter the line command "DE" to delete a procedure.

NDB - Interactive SQL

The Interactive SQL function of the Natural Tools for DB2 enables you to execute SQL statements dynamically.

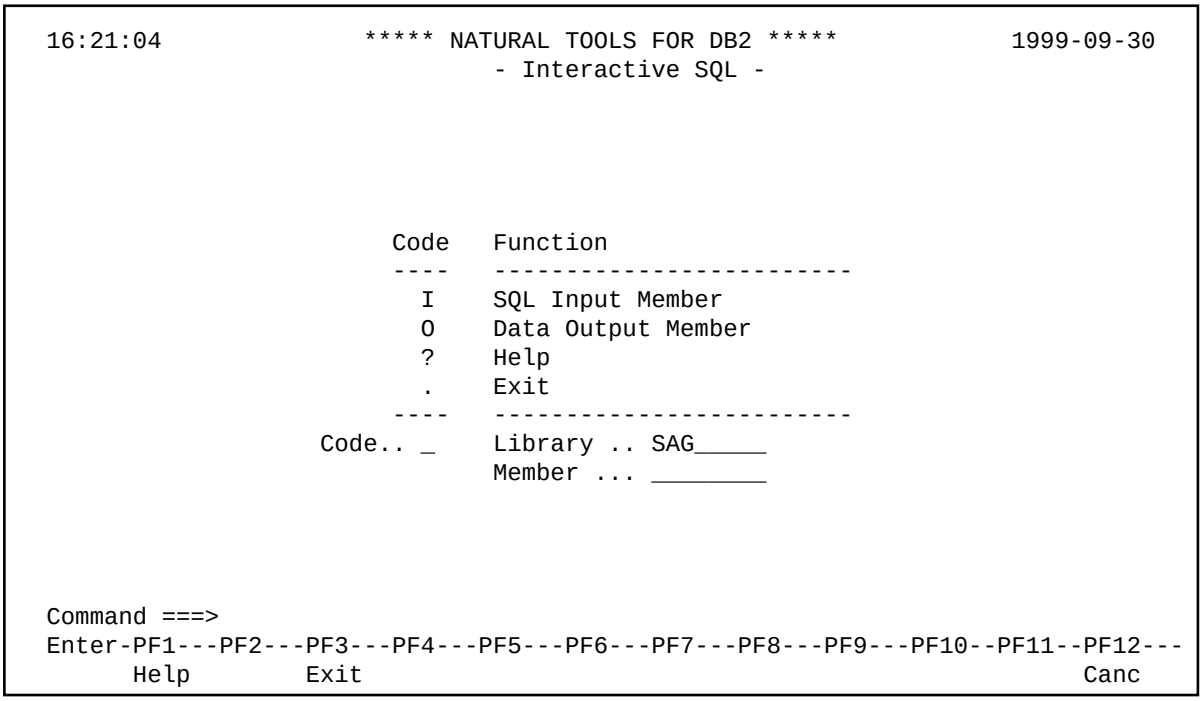
- Invoking the Interactive SQL Function
- SQL Input Members
- Data Output Members
- Processing SQL Statements
- PF-Key Settings
- Unloading Interactive SQL Results

Invoking the Interactive SQL Function

To invoke the Interactive SQL function

- Enter function code "I" on the Natural Tools for DB2 Main Menu.

The Interactive SQL screen is displayed:



The following functions are available:

Code	Description
I	Displays SQL members in the interactive SQL input screen.
O	Displays output members in the interactive SQL output screen.

The following parameters can be specified:

Parameter	Description
Library	Specifies the name of the current Natural library which contains the specified input/output members. Specification of libraries whose names begin with "SYS" is not allowed. The library name is preset with your Natural user ID.
Member	If a valid member name is specified, the corresponding member is displayed. If a value is specified followed by an asterisk (*), all input/output members in the current library whose names begin with this value are listed. If asterisk notation is specified only, a selection list of all input/output members in the current library is displayed. If the Member field is left blank, the empty SQL input/output screen is displayed.

SQL Input Members

To invoke the SQL Input Member function enter function code "I" on the Interactive SQL screen. Depending on what member name you have specified, different screens are displayed.

SQL Input Screen

If you leave the Member field blank, the empty SQL input screen is invoked:

```
16:21:56              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
 ISQL - Input        SAG                  S 01- ---------------Columns 001 072
 ====>                                                        Scroll ===>  PAGE
 ***** ***************************** top of data ******************************
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ,,,,,
 ***** ***************************** bottom of data ***************************

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Setup Exit  Exec  Rfind Rchan -     +     Outpu             Canc
```

The SQL input screen is a free-mode editor which provides a functionality similar to the one of the Software AG Editor. Using the editor you can enter or edit SQL statements via editor main and line commands. You can execute the SQL statements immediately from within the editor by pressing PF4 (Exec), or you can save them as an SQL member in a Natural library for later execution.

Note:
The PRINT command is not available in the SQL input screen.

Apart from the editor main and line commands, SQL code maintenance commands are also available to maintain SQL members in a Natural library. With these maintenance commands, input members can be listed, retrieved, saved in a Natural library, copied, and purged. They are entered in the command line of the input screen.

You can also obtain a list of the available maintenance commands by entering the help character "?" in the command line of the input screen. A window is displayed from which the desired command can be selected. The window can be scrolled forwards by pressing PF8, or backwards by pressing PF7.

```
10:57:06              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
 ISQL - Input        SAG                  S 01- ---------------Columns 001 072
 ====> ?                                                      Scroll ===>  PAGE
 ***** ********************** +-----------------------------------+*************
                              !                                   !
                              !   _  List <*,member>              !
                              !   _  READ <member>                !
                              !   _  SAVE <member>                !
                              !   _  COPY <member>                !
                              !   _  Purge <member>               !
                              !   _  LIBrary <library>            !
                              !   _  SELect <TB,CO> name1 name2   !
                              !                                   !
                              +-----------------------------------+




 ***** **************************** bottom of data ****************************

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Setup Exit  Exec  Rfind Rchan -     +     Outpu             Canc
```

To assist you in coding your SQL member, exisiting DB2 tables and columns can be listed using the SELECT command. From the list, you can include table and column names into the editor.

The SELECT command is available for table and column selection:

Command	Description
SELECT TABLE [*creator*.]*name*	Selects all tables with the specified creator (optional) and name. For both *creator* and *name*, you can specify a value followed by an asterisk (*), and all tables whose names begin with this value are selected. If you specify asterisk notation only, all existing tables are selected. If you specify a table name without a creator, all tables with the specified name are selected, regardless of their creator.
SELECT COLUMN *creator*.*name*	Selects all columns of the table "*creator*.*name*". Since the table must be uniquely identified, asterisk notation cannot be used.

Sample Input Screen with Table Listing Window

```
11:02:50              ***** NATURAL Tools F +--------------------------------+
 ISQL - Input        SAG                  S !  Tab:                          !
 ====>                                     !    SYSIBM.*                      !
 ***** ****************************** top of !    Table Name         Creator  !
                                           !  _ SYSPLANAUTH        SYSIBM   !
                                           !  _ SYSPLANDEP         SYSIBM   !
                                           !  _ SYSRELS            SYSIBM   !
                                           !  _ SYSRESAUTH         SYSIBM   !
                                           !  _ SYSSTMT            SYSIBM   !
                                           !  _ SYSSTOGROUP        SYSIBM   !
                                           !  _ SYSSYNONYMS        SYSIBM   !
                                           !  _ SYSTABAUTH         SYSIBM   !
                                           !  _ SYSTABLEPART       SYSIBM   !
                                           !  _ SYSTABLES          SYSIBM   !
                                           !  _ SYSTABLESPACE      SYSIBM   !
                                           !  _ SYSUSEROUT         SYSIBM   !
                                           !  _ SYSUSERNAMES       SYSIBM   !
                                           !  _ SYSVUIEWDEP        SYSIBM   !
                                           !  _ SYSVIEWS           SYSIBM   !
                                           !  _ SYSVLTREE          SYSIBM   !
 ***** *************************** bottom o !  _ SYSVOLUMES         SYSIBM   !
                                           !  _ SYSVTREE           SYSIBM   !
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF !                               !
      Help  Setup Exit  Exec  Rfind Rchan  - +--------------------------------+
```

From the table list, you can select a table for display of its columns by marking it with "C" in front of the table name. The columns of a table are listed together with their type and length.

Sample Input Screen with Column Listing Window

```
11:02:50               ***** NATURAL  +-------------------------------------+
 ISQL - Input        SAG                !  Tab: SYSIBM.SYSTABLES             !
 ====>                                  !                                    !
 ***** ***************************** t !    Column Name        Type     Len  !
 A     SELECT                           !  M NAME               VARCHAR  18   !
 00002 FROM SYSIBM.SYSTABLES;           !  M CREATOR            CHAR     8    !
 ***** *************************** bot !  M TYPE               CHAR     1    !
                                        !  M DBNAME             CHAR     8    !
                                        !  M TSNAME             CHAR     8    !
                                        !  _ DBID               SMALLINT 2    !
                                        !  _ OBID               SMALLINT 2    !
                                        !  _ COLCOUNT           SMALLINT 2    !
                                        !  _ EDPROC             CHAR     8    !
                                        !  _ VALPROC            CHAR     8    !
                                        !  _ CLUSTERTYPE        CHAR     1    !
                                        !  _ CLUSTERRID         INTEGER  4    !
                                        !  _ CARD               INTEGER  4    !
                                        !  _ NPAGES             INTEGER  4    !
                                        !  _ PCTPAGES           SMALLINT 2    !
                                        !  _ IBMREQD            CHAR     1    !
                                        !  _ REMARKS            VARCHAR  254  !
                                        !  _ PARENTS            SMALLINT 2    !
Enter-PF1---PF2---PF3---PF4---PF5---PF6 !                                    !
      Help  Setup Exit  Exec  Rfind Rch +-------------------------------------+
```


If you want to copy table or column names from a selection list into the editor, mark the corresponding table or column with "M" as shown on the previous screen. The table or column names are copied either after or before the line marked with "A" or "B" respectively, or to the top of the displayed data.

Sample Input Screen with Copied Column Names

```
11:02:50                ***** NATURAL  +-------------------------------------+
 ISQL - Input        SAG                !  Tab: SYSIBM.SYSTABLES              !
 ====>                                  !                                     !
 ***** **************************** t  !    Column Name        Type     Len  !
 A     SELECT                           !  _ NAME               VARCHAR  18   !
 00002 NAME                             !  _ CREATOR            CHAR     8    !
 00003 , CREATOR                        !  _ TYPE               CHAR     1    !
 00004 , TYPE                           !  _ DBNAME             CHAR     8    !
 00005 , DBNAME                         !  _ TSNAME             CHAR     8    !
 00006 , TSNAME                         !  _ DBID               SMALLINT 2    !
 00007 FROM SYSIBM.SYSTABLES;           !  _ OBID               SMALLINT 2    !
 ***** ************************** bot  !  _ COLCOUNT           SMALLINT 2    !
                                        !  _ EDPROC             CHAR     8    !
                                        !  _ VALPROC            CHAR     8    !
                                        !  _ CLUSTERTYPE        CHAR     1    !
                                        !  _ CLUSTERRID         INTEGER  4    !
                                        !  _ CARD               INTEGER  4    !
                                        !  _ NPAGES             INTEGER  4    !
                                        !  _ PCTPAGES           SMALLINT 2    !
                                        !  _ IBMREQD            CHAR     1    !
                                        !  _ REMARKS            VARCHAR  254  !
                                        !  _ PARENTS            SMALLINT 2    !
Enter-PF1---PF2---PF3---PF4---PF5---PF6 !                                     !
      Help  Setup Exit  Exec  Rfind Rch +-------------------------------------+
```


Fixed Mode with Interactive SQL

All fixed-mode input screens from the Catalog Maintenance part of the Natural Tools for DB2 are available as help maps within the Interactive SQL part. To invoke this help facility, enter the name of the SQL statement you want to create in the command line of your SQL input screen, for example, "CREATE TABLE" or "CR TB" for the CREATE TABLE command.

The same command abbreviations apply as with the Catalog Maintenance.

If you enter "CREATE TABLE" or "CR TB", the Create Table screen is invoked:

```
  09:47:19              ***** NATURAL TOOLS FOR DB2 *****             2002-08-12
                                  - Create Table -                       1  / 9

 >>- CREATE TABLE - SAG_____ . DEMOTABLE_________ ----------------------------->
                     <creator.>table-name
 >+--- LIKE ------- ________ . ___________________ +-------------------------+-+>
  !                  <creator.>table/view-name    +- _ - INCLUDING IDENTITY + +
  !                                                                           !
  +( COL1_____________  CHAR___________ ( 20________ ) _ - __ - _ - __ - _ , +
  +- COL2_____________  INTEGER________ ( ___________ ) _ - NN - _ - 2_ - _ , +
  +- COL3_____________  SMALLINT_______ ( ___________ ) _ - NN - _ - 1_ - _ , +
  +- COL4_____________  CHAR___________ ( 2_________ ) S - __ - _ - __ - _ , +
  +- COL5_____________  VARCHAR________ ( 30________ ) _ - NN - _ - 3_ - _ , +
  +- COL6_____________  DECIMAL________ ( 2,5_______ ) _ - __ - X - __ - _ , +
  +- COL7_____________  FLOAT__________ ( ___________ ) _ - NN - _ - __ - _ , +
  +- COL8_____________  DATE___________ ( ___________ ) _ - __ - _ - __ - _ , +
  +- COL9_____________  TIME___________ ( ___________ ) _ - __ - _ - __ - _ , +
  +- _________________  _______________ ( __________ ) _ - __ - _ - __ - _ , +
        column-name          format           length    S/M  NN  fld  PK/ R/C
                                                         B       proc UK  D/G
 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec  Free  --     -     +    ++          Next  Canc
```

If you have entered data for a complete SQL statement, you can generate an SQL statement from the entered data and include it into the SQL input screen. Using PF4 (Incl), you include the generated SQL code and remain on the Create Table screen. Using PF5 (IBack), you include the generated SQL code and return to the SQL input screen.

Retrieve an SQL Member

If you specify a unique member name in the Member field of the Interactive SQL screen, the corresponding SQL member is listed on the input screen. If no member exists with the specified name, a corresponding message is returned.

Sample SQL Member Listed in Input Screen

```
16:27:12              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
 ISQL - Input        SAG(TESTSEQ)          S 01- ---------------Columns 001 072
 ====>                                                 Scroll ===>  PAGE
 ***** **************************** top of data ****************************
 00001 CREATE TABLE DEMOTABLE
 00002   (COL1                 CHAR(8),
 00003    COL2                 INTEGER
 00004   ) IN DATABASE DEMO;
 00005 INSERT INTO DEMOTABLE
 00006   VALUES ('AAAAA',1);
 00007 * INSERT INTO DEMOTABLE
 00008 *   VALUES ('BBBBB',2);
 00009 SELECT FROM DEMOTABLE;
 00010 DROP TABLE DEMOTABLE;
 ***** ************************** bottom of data ****************************




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Setup Exit  Exec  Rfind Rchan -     +     Outpu             Canc
```

Listed SQL members can be purged, modified, executed, or saved.

An asterisk (*) in front of a statement line turns this line into a comment line, which means that the corresponding SQL code is not considered for execution.

List of SQL Members

If you specify a value followed by an asterisk (*) in the Member field, a list of all SQL input members in the current library whose names begin with this value is displayed.

If you specify an asterisk (*), a list of all SQL input members in the current library is displayed.

Sample SQL Input Member Selection List

```
 13:57:15              ***** NATURAL TOOLS FOR DB2 *****             2002-08-12
                                Select Member

       C      Member      Type         User        Date        Time
       -      --------    -----------  --------    ----------  --------
       _      CRAXTB      SQL          SAG         2002-02-26  10:48:53
       _      CRDITY      SQL          SAG         2002-02-22  15:39:14
       _      CRPRQE      SQL          SAG         2002-08-12  13:54:21
       _      CRTB        SQL          SAG         2001-12-20  09:48:14
       _      CRTRIG      SQL          SAG         2002-02-26  15:53:01
       _      CRTRIG2     SQL          SAG         2002-02-26  09:14:10
       _      DRPRQE      SQL          SAG         2002-08-12  13:55:04
       _      DRPRQE2     SQL          SAG         2002-05-22  16:50:30
       _      GGSDTYPE    SQL          SAG         2002-08-12  13:52:10
       _      GRSHPR      SQL          SAG         2002-02-26  16:28:01
       _      RESHPR      SQL          SAG         2002-02-26  16:31:05
       _      SELPROCS    SQL          SAG         2002-05-29  16:09:05
       _      SELTABS     SQL          SAG         2002-08-12  13:56:22



Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Cont              Exit                                           >     Canc
```

From the input screen selection list, SQL members can be selected for display by marking them with an "S". If the list has been invoked by a PURGE command, members can be purged by marking them with a "P".

By pressing PF11, you can switch from the default view of the Select Member screen as shown above to the extended view with the first line of each member displayed in the Description column:


```
14:14:20              ***** NATURAL TOOLS FOR DB2 *****            2002-08-12
                                Select Member

      C      Member      Description (first line of member)
      -      --------    ------------------------------------------------
      _      CRAXTB      CREATE AUXILIARY TABLE aux-table-name
      _      CRDITY      CREATE DISTINCT TYPE distinct-type-name
      _      CRPRQE      * ALL PROCEDURES FROM QARNDB31(10,110), WHICH HAVE 'C
      _      CRTB        CREATE TABLE NEWTYPE
      _      CRTRIG      CREATE TRIGGER trigger-name NO CASCADE BEFORE|
      _      CRTRIG2     CREATE TRIGGER trigger-name (NO CASCADE BEFORE|
      _      DRPRQE      * ALL PROCEDURES FROM QARNDB31(10,110), WHICH HAVE 'C
      _      DRPRQE2     DROP PROCEDURE CALLN2 RESTRICT;
      _      GGSDTYPE    SELECT COLTYPE,LENGTH,LENGTH2,DATATYPEID,SOURCETYPEID
      _      GRSHPR      GRANT ALTERIN [, CREATEIN] [, DROPIN]
      _      RESHPR      REVOKE ALTERIN [, CREATEIN] [, DROPIN]
      _      SELPROCS    SELECT * FROM SYSIBM.SYSPROCEDURES
      _      SELTABS     SELECT * FROM SYSIBM.SYSTABLES



Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Cont              Exit                                    <           Canc
```

The first line of a member can be the first line of an SQL statement or a comment line which provides more information on the member.

Data Output Members

To invoke the Data Output Member function, enter function code "O" on the Interactive SQL screen. Depending on what member name you have specified, different screens are displayed.

Data Output Screen

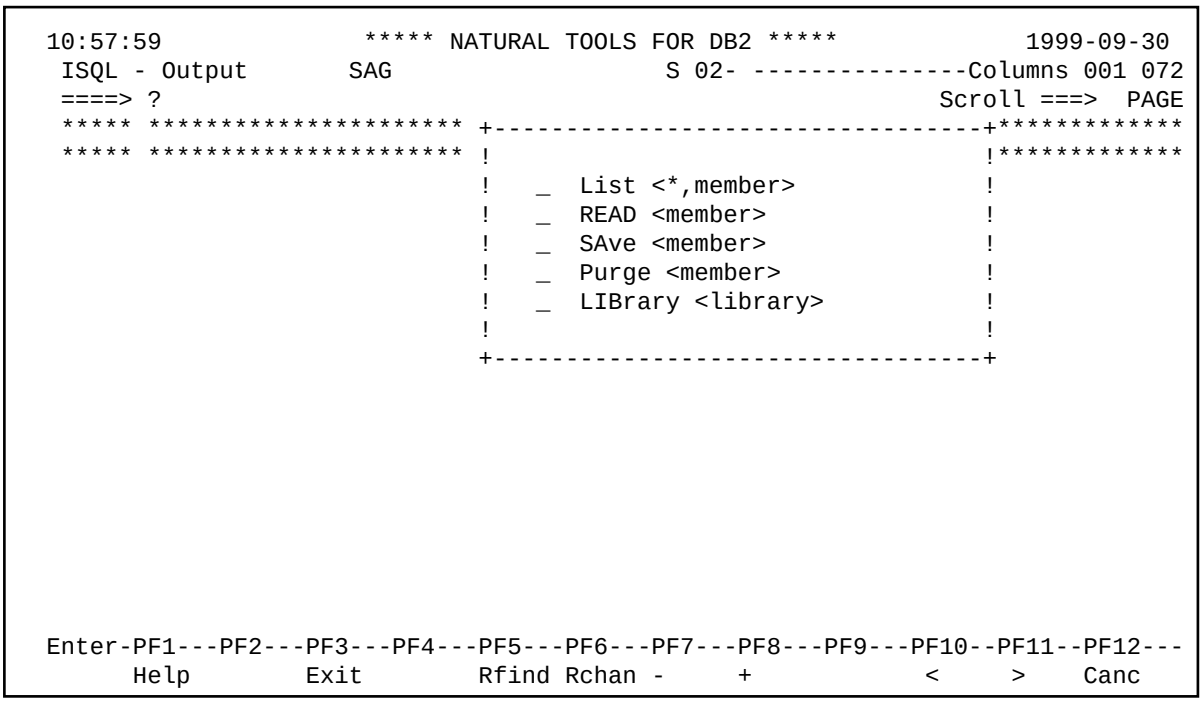
If you leave Member field blank, the empty SQL output screen is invoked.

```
16:29:35              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
 ISQL - Output       SAG                   S 02- ---------------Columns 001 072
 ====>                                                          Scroll ===>  PAGE
 ***** ***************************** top of data ****************************
 ***** *************************** bottom of data ***************************









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind Rchan -     +           <     >    Canc
```

From the data output screen you have access to output data members only. Output members consist of data retrieved from the database as a result of executed SQL statements. These data can be browsed and saved for later use as output members on the Natural system file (FUSER). In addition to the data retrieved from the database, output members also contain DB2 status information, and the executed SQL member.

If you execute an SQL statement, the results are automatically shown on the output screen. Thus, you can enter the interactive SQL output screen also by executing an SQL statement from the input screen. From the output screen you can return to the input screen by pressing PF3 (Exit).

The maintenance commands available for output members can be displayed and selected in a window, too. The window is invoked by entering the help character "?" in the command line of the output screen.



Apart from the maintenance commands, only browse commands are available, since output members cannot be modified. Both browse and maintenance commands are entered in the command line of the output screen.

If an output member is too large to fit on your terminal screen, you can use the FIX ON *n* command to keep the first *n* characters on the screen when scrolling to the left or to the right.

Retrieve an Output Member

If you specify a unique member name in the Member field of the Interactive SQL screen, the corresponding output member is listed on the output screen. If no member exists with the specified name, a corresponding message is returned.

Sample Output Member Listed in Output Screen

```
16:27:12              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
 ISQL - Output       SAG(TESTSEQO)         S 02- ---------------Columns 001 072
 ====>                                                          Scroll ===>  PAGE
 ***** **************************** top of data ****************************
 00001 CREATE TABLE DEMOTABLE
 00002   (COL1                CHAR(8),
 00003    COL2                INTEGER
 00004   ) IN DATABASE DEMO
 00005 ------------------------------------------------------------------------
 00006 STATEMENT WAS SUCCESSFUL, SQLCODE = 0
 00007 ------------------------------------------------------------------------
 00008 INSERT INTO DEMOTABLE
 00009   VALUES ('AAAAA',1)
 00010 ------------------------------------------------------------------------
 00011 STATEMENT WAS SUCCESSFUL, SQLCODE = 0
 00012 ------------------------------------------------------------------------
 00013 SELECT FROM DEMOTABLE
 00014 ------------------------------------------------------------------------
 00015 COL1          COL2
 00016 ------------------------------------------------------------------------
 00017 AAAAA            1

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind Rchan -     +           <     >    Canc
```


List of Output Members

If you specify a value followed by an asterisk (*) in the Member field of the Interactive SQL screen, a list of all data output members in the current library whose names begin with this value is displayed.

If you specify asterisk notation only, a list of all data output members in the current library is displayed.

Sample Data Output Member Selection List

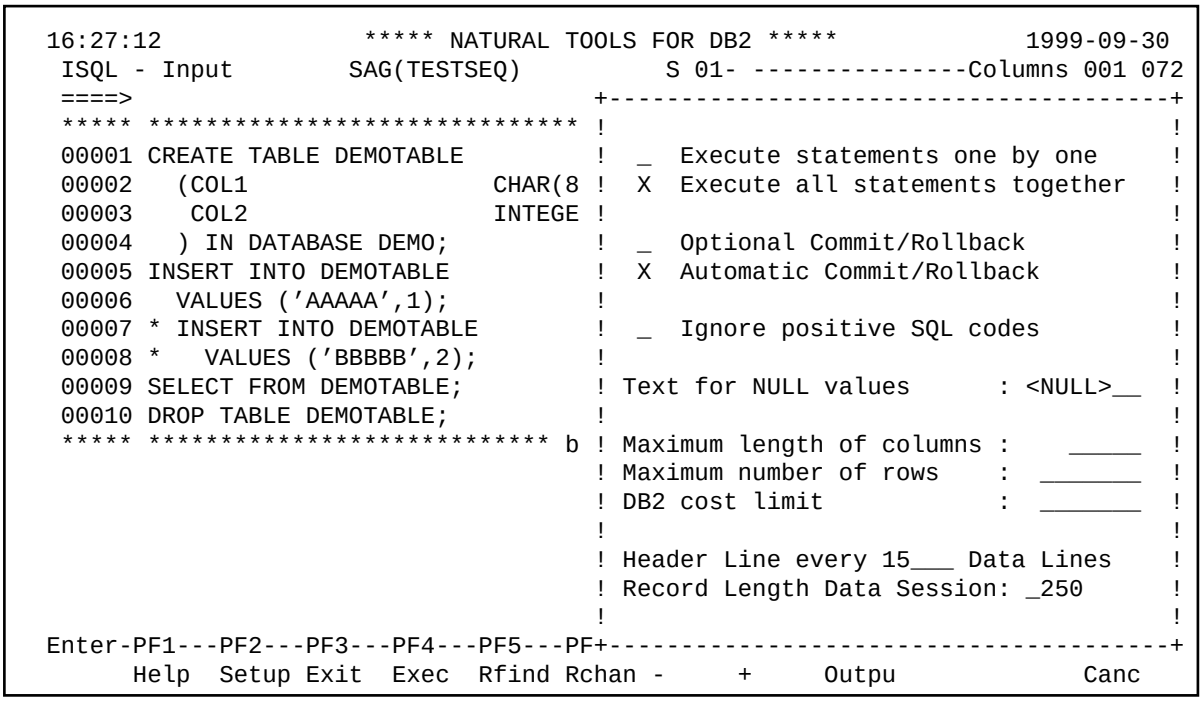
```
16:24:02              ***** NATURAL TOOLS FOR DB2 *****           2002-08-29
                                Select Member


      C     Member      Type         User        Date        Time
      -     --------    -----------  --------    --------    -----------
      _     AAAA        SQL-RESULT   SAG         1997-01-18  13:54:54
      _     ADEMVIEW    SQL-RESULT   SAG         1997-10-23  14:01:09
      _     AIRCRAFT    SQL-RESULT   SAG         1997-11-28  10:01:32
      _     BBBB        SQL-RESULT   SAG         1997-01-18  15:25:14
      _     BSP1        SQL-RESULT   SAG         1997-01-18  14:57:11
```

From the output member selection list, output members can be selected for display by marking them with an "S". If the list has been invoked by a PURGE command, members can be purged by marking them with a "P".

Processing SQL Statements

SQL input members can only be accessed from the input screen. They are executed from the input screen against DB2 by pressing PF4 (Exec). After execution, the data output screen appears which contains the results of the executed SQL member.

If an SQL member consists of more than one SQL statements, the individual statements must be separated by a semicolon. They can be executed one by one or all together at the same time. To choose the form of execution, a window is provided which can be invoked by pressing PF2 (Setup).



Below is information on the options provided:

- Execute Statements One By One
- Execute All Statements Together
- Automatic Commit/Rollback
- Optional Commit/Rollback
- Text For NULL Values
- Maximum Length of Columns
- Maximum Number of Rows
- DB2 Cost Limit
- Header Line Every *n* Data Lines
- Record Length Data Session

Execute Statements One By One

After each SQL statement the output screen is shown. From the output screen, you can either execute the next SQL statement from the input screen by pressing PF4 (Next), or skip the remaining SQL statements and return to the input screen immediately by pressing PF3 (Exit).

Execute All Statements Together

All statements are executed immediately one after the other. The output screen shows the results of all statements together.

Statements containing cursor names, host variables, or parameter markers cannot be executed with interactive SQL. Also not executed are statements available as embedded SQL only; that is, statements whose functions are automatically performed by Natural.

These statements are:

CLOSE
CONNECT
DECLARE
DELETE WHERE CURRENT OF CURSOR
DESCRIBE
EXECUTE
FETCH
INCLUDE
OPEN
PREPARE
SELECT INTO
SET *host-variable*
SET CURRENT PACKAGESET
UPDATE WHERE CURRENT OF CURSOR
WHENEVER

Automatic Commit/Rollback

If you select automatic commit/rollback, each modification of the database is automatically either committed or rolled back, depending on whether all the SQL statements involved execute successfully. If so, an SQL COMMIT WORK command is executed; if not, an SQL ROLLBACK command backs out all database modifications since the last commit point.

Optional Commit/Rollback

If you select optional commit/rollback, a window is invoked after each SQL statement, offering you the option to either commit or roll back the resulting database modifications shown on the screen.

Note:
Since under CICS and IMS/TM each terminal I/O results in a SYNCPOINT, the optional commit/rollback feature only applies in a TSO environment.

In all environments, you can include SQL COMMIT and ROLLBACK commands in your input member, too. Under CICS and IMS/TM, however, these commands are translated into the corresponding TP-monitor calls.

Text For NULL Values

The text that is to be shown for NULL values can be specified here; the default string is '---'.

Maximum Length of Columns

Limits the length for a single column to *n* characters. This limit only applies to character data. DATE, TIME, or NUMERIC columns are not truncated. The value 0 indicates that no limit exists.

Maximum Number of Rows

Limits the number of rows returned by one SELECT statement. The value 0 indicates that no limit exists.

DB2 Cost Limit

Sets a limit for the DB2 cost estimate. SELECT statements which exceed this limit are not executed. The value 0 indicates that no limit exists.

Header Line Every *n* Data Lines

For SELECT statements, you can specify that every *n* data lines a header line is inserted with the names of the selected columns. If *n* is set to 0, only one header line is displayed at the top of the data.

Record Length Data Session

The record length (*n*) for the output session can be specified. If the specified record length is smaller than the record length of the output data, the output records are truncated accordingly. The truncation of records is indicated by a greater than character (>) as the leftmost character in the first line beneath each header line. The default value for *n* is 250 bytes.

PF-Key Settings

The following PF-key settings apply to the input screen:

Key	Setting	Function
PF2	Setup	Invokes a window with further processing options.
PF4	Exec	Executes the SQL member currently in the input screen.
PF5	Rfind	Repeats the last executed FIND command.
PF6	Rchan	Repeats the last executed CHANGE command.
PF7	-	Scrolls the display one page backward.
PF8	+	Scrolls the display one page forward.
PF9	Outpu	Invokes the output member selection list directly from within the input screen.

Apart from PF2 (Setup), PF4 (Exec), and PF9 (Outpu), the same PF-key settings apply to the output screen, too. In addition, the following PF-key settings are available:

Key	Setting	Function
PF4	Next	Executes the next SQL statement if an SQL member consists of more than one statement, and if you have chosen to execute them one after the other. If not, the setting for PF4 is left blank.
PF10	<	Scrolls the display of the output screen to the left.
PF11	>	Scrolls the display of the output screen to the right.

Unloading Interactive SQL Results

Results from interactive SQL are unloaded and written to a dataset referred to by DD name CMWKF01 in batch mode using the UNLDDATA command.

CMWKF01 should be of variable record format; the record length depends on the size of the SQL output member and can range from 250 to 4000 bytes.

If UNLDDATA is issued in the Natural system library SYSDB2, the Unload SQL Results menu is displayed:

```
16:53:20              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
                             - Unload SQL Results -




                           Code Function
                           ---- -------------------
                            U   Unload SQL Results
                            .   Exit
                           ---- -------------------
                     Code .. _   Library .. _________
                                 Member ... _________



Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                   Exit                                                  Canc
```

The following function is available:

Code	Description
U	Unloads results from interactive SQL execution.

The following parameters apply:

Parameter	Description
Library	Specifies the name of the Natural library from which the specified output members are to be unloaded. You cannot specify libraries whose names begin with "SYS". This parameter must be specified.
Member	Specifies the name(s) of the output member(s) to be unloaded. This parameter must be specified.

NDB - Retrieval of System Tables

Important:
Before you use the Retrieval of System Tables function, refer to LISTSQL and Explain Functions in the section Special Requirements for Natural Tools for DB2.

The DB2 system tables provide information on the contents of your DB2 system. The Retrieval of System Tables function enables you to:

- display information on DB2 objects without coding SQL queries;
- easily access related objects, such as indexes of a table.

The DB2 objects supported by the Retrieval of System Tables function are database, tablespace, table, index, column, plan, check constraints, statistic tables, package, and DBRM (database request module), as well as access rights to and relationships between these objects.

DB2 objects are presented in one of the following two ways:

- As selection lists, where all objects are of the same type, and where commands can be issued to display related objects.
- You can list databases, tables, plans, and packages by name. From the database listings, you can invoke listings of the tablespaces or tables of a database. From the table listing, you can invoke listings of the columns and indexes of a table. From the plan listing, you can invoke listings of the DBRMs of a plan, of the package list of a plan, of the tables and indexes used by a plan, and of the systems which are enabled or disabled for a plan. From the package listing, you can invoke listings of the tables and indexes used in a package and of the systems which are enabled or disabled for a package. From the database, table, plan, or package listings, you can also investigate who is authorized to access a DB2 object. In addition, the User Authorization menu enables you to list all existing access rights by user ID.
- As reports, which merely contain information on different types of DB2 objects, and where only browse commands can be issued.

Browsing of objects is performed with ISPF-like commands. The most important browse commands can also be issued via PF keys.

This section covers the following topics:

- Invoking the Retrieval of System Tables Function
- List Databases
- List Tablespaces
- List Plans
- List Packages
- List Tables
- User Authorizations
- List Statistic Tables

Invoking the Retrieval of System Tables Function

To invoke the Retrieval of System Tables function

- Enter function code "R" on the Natural Tools for DB2 Main Menu.

The Retrieval of System Tables screen is displayed:

```
16:31:56              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
                          - Retrieval of System Tables -

                     Code  Function            Parameter

                      D    List Databases      Database
                      K    List Packages       Collection, Name
                      P    List Plans          Plan
                      T    List Tables         Tbreator, Tbname
                      U    User Authorizations
                      S    Statistic Tables
                      ?    Help
                      .    Exit

               Code .. _    Database Name ....... _________
                            Package Collection .. ____________________
                            Package Name ........ _________
                            Plan Name ........... _________
                            Table Creator ....... _________
                            Table Name .......... ____________________

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                  Canc
```

The following functions are available:

Code	Description
D	Lists databases defined in the DB2 catalog.
K	Lists packages defined in the DB2 catalog.
P	Lists plans defined in the DB2 catalog.
S	Statistic tables.
T	Lists tables defined in the DB2 catalog.
U	Provides information on which user(s) can access which DB2 objects.

The following parameters must be specified as selection criteria:

Parameter	Description
Database Name	The name of the database to be listed. Asterisk notation (*) for range specification is possible. The Database Name parameter is relevant to the List Databases function only.
Package Collection	The collection of the package to be listed. Asterisk notation (*) for range specification is possible. The Package Collection parameter is relevant to the List Packages function only.
Package Name	The name of the package to be listed. Asterisk notation (*) for range specification is possible. The Package Name parameter is relevant to the List Packages function only.
Plan Name	The name of the plan to be listed. Asterisk notation (*) for range specification is possible. The Plan Name parameter is relevant to the List Plans function only.
Table Creator	The name of the creator of the table(s) to be listed. Asterisk notation (*) for range specification is possible. The Table Creator parameter is relevant to the List Tables function only.
Table Name	The name of the table to be listed. Asterisk notation (*) for range specification is possible. The Table Name parameter is relevant to the List Tables function only.

List Databases

To invoke the List Databases function

1. Enter function code "D" on the Retrieval of System Tables screen.
2. Specify the name of the database(s) to be listed.

If a value followed by an asterisk is specified, all databases defined in the DB2 catalog whose names begin with this value are listed.

If asterisk notation is specified only, all databases defined in the DB2 catalog are listed.

```
16:32:24              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
 DATABASES *                                S 01     Row 0 of 25 Columns 001 059
 ====>                                                          Scroll ===>  PAGE
    DATABASE CREATOR     STOGROUP BPOOL     DBID CREATEDBY ROSHARE TIMESTAMP
 ** ******************************** top of data ********************************
 __ DEMO     DEFAULT     SYSDEFLT BP0        269  DEFAULT           0001-01-01-0
 __ DEMODB   SAG2        SYSDEFLT BP0        273  SAG2              0001-01-01-0
 __ DEVELOP  SAG         DEVELOP  BP0        260  SAG               0001-01-01-0
 __ ECHDB01  SAG2        SYSDEFLT BP0        272  SAG2              0001-01-01-0
 __ EFGDB    SAG         SYSDEFLT BP0        263  SAG               0001-01-01-0
 __ HBUTST   SAG2        SYSDEFLT BP0        275  SAG2              0001-01-01-0
 __ PLANTAB  SAG2        SYSDEFLT BP0        270  SAG2              0001-01-01-0
 __ Predict  SAG2        SYSDEFLT BP0        262  SAG2              0001-01-01-0
 __ QA       SAG2        SYSDEFLT BP0        265  SAG2              0001-01-01-0
 __ SAGDB04  SYSIBM      SYSDEFLT BP0          4  SYSIBM            0001-01-01-0
 __ SAGDB06  SYSIBM                            6  SYSIBM            0001-01-01-0
 __ SAGDB07  SAG1        SYSDEFLT BP0          7  SAG1              0001-01-01-0
 __ SAGDDF   SAG1        SYSDEFLT BP0        257  SAG1              0001-01-01-0
 __ SAGRLST  SAG1        SYSDEFLT BP0        256  SAG1              0001-01-01-0
 __ SAG8D22A SAG1        SAG8G220 BP0        258  SAG1              0001-01-01-0
 __ SAG8D22P SAG1        SAG8G220 BP0        259  SAG1              0001-01-01-0

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +          <      >   Canc
```

Commands Allowed on Databases

The following line commands are available on the database listing screen. Line commands are entered in front of the desired database(s):

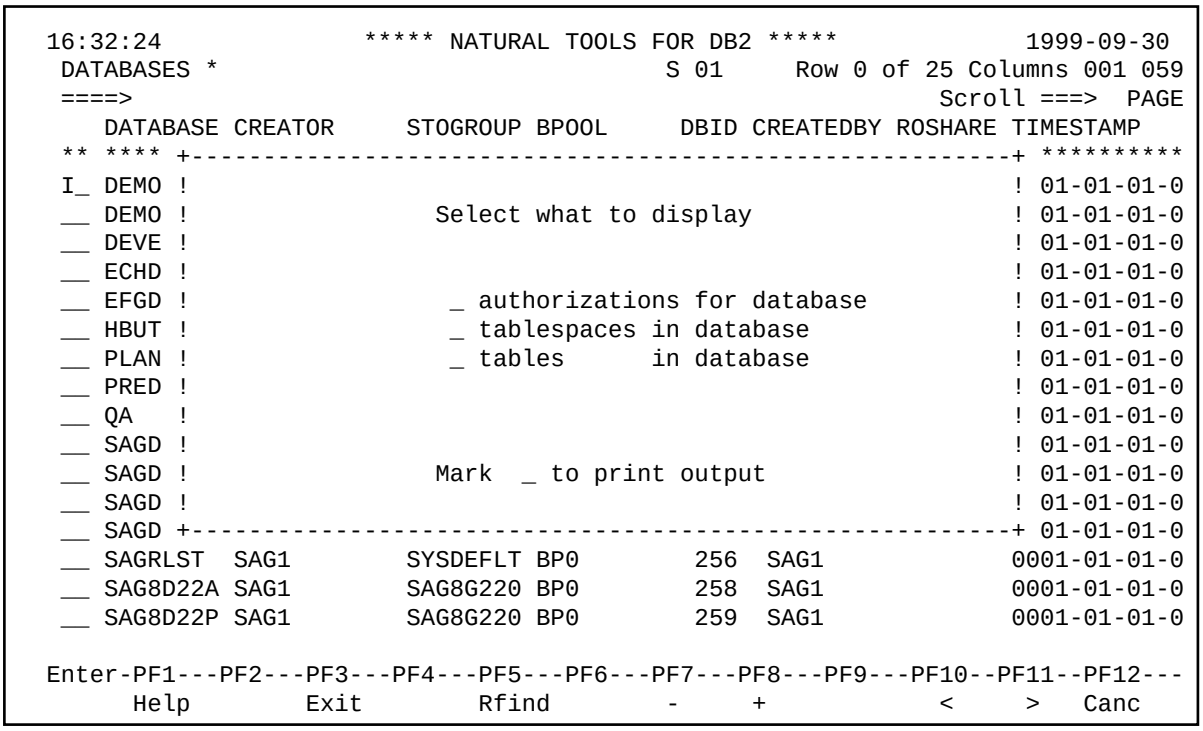
Command	Description
I	Displays information on a database.
S	Selects a database to be used with main commands (see below).
U	Unselects a database.
AU	Displays information on access rights to a database.
TB	Displays all tables defined in a database.
TS	Displays all tablespaces defined in a database.

The listings of tables or tablespaces displayed as a result of the "TB" or "TS" command can be used for further processing, whereas the contents of the screens displayed as a result of the "AU" or "I" command are for information purposes only.

A list of all line commands available with the List Database function can be invoked as a window by entering the help character "?" in front of any of the listed databases.

The commands "AU", "TB", and "TS" can also be used as main commands. Main commands are entered in the command line of the database list screen and apply to all databases previously selected with the "S" line command.

A further main command is the INFO command, which is the equivalent of the "I" line command, but displays information on all previously selected databases. Instead of being displayed, all information resulting from the "I" or INFO commands can also be marked for printing. Even if already displayed, information can be printed by issuing the PRINT command.



A list of all main commands available with the List Database function can be invoked as a window by entering "?" in the command line of the database list screen.

List Tablespaces

The function to list tablespaces is not part of the Retrieval of System Tables main menu.

To list tablespaces

- Issue the "TS" command on the database listing screen only.

A tablespace listing screen is displayed.

```
16:35:07              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
 TABLESPACES IN DATABASE DB2DEMO           S 02      Row 0 of 2 Columns 032 075
 ====>                                                 Scroll ===>  PAGE
    DATABASE NAME        CREATOR  BPOOL    PGSIZE PARTITIONS NTABLES SEGSIZE LO
 ** ******************************* top of data ********************************
 __ DB2DEMO  AUTOMOBI    SAG      BP0           4          0       1       0 A
 __ DB2DEMO  EMPLOYEE    SAG      BP0           4          0       1       0 A
 ** ***************************** bottom of data ******************************
```

Commands Allowed on Tablespaces

The following line commands are available on the tablespace listing screen. Line commands are entered in front of the desired tablespace(s):

Command	Description
I	Displays information on a tablespace.
S	Selects a tablespace to be used with main commands.
U	Unselects a tablespace.
PT	Displays all partitions of a tablespace.
TB	Displays all tables defined in a tablespace.

The listings of tables displayed as a result of the "TB" command can be used for further processing, whereas the listings resulting from the "I" and "PT" commands are for information purposes only.

A list of all line commands available on the tablespace listing screen can be invoked as a window by entering the help character "?" in front of any of the listed tablespaces.

The commands "PT" and "TB" can also be used as a main commands entered on the command line of the tablespace listing screen. Main commands apply to all tablespaces previously selected with the "S" line command.

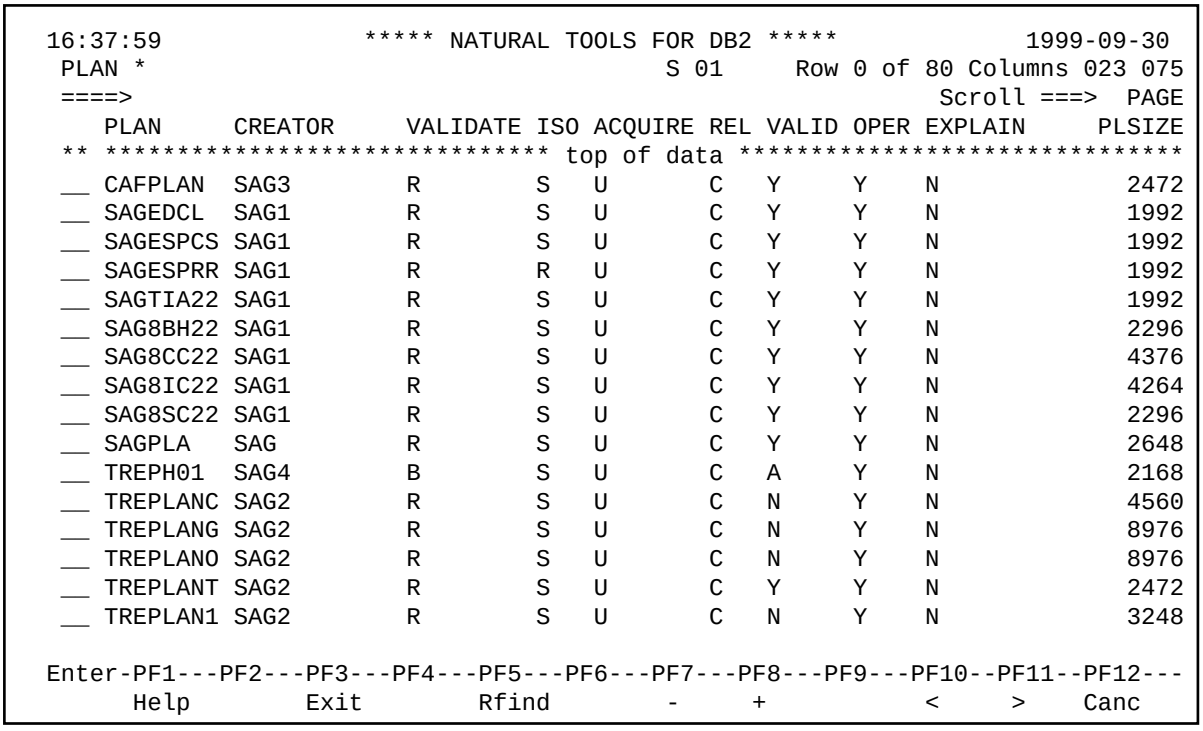
A further main command is the INFO command, which is the equivalent of the "I" line command, but displays information on all previously selected tablespaces. Instead of being displayed, all information resulting from the "I" or INFO commands can also be marked for printing. Even if already displayed, information can be printed by issuing the PRINT command.

```
16:35:07              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
 TABLESPACES IN DATABASE DB2DEMO           S 02      Row 0 of 2 Columns 032 075
 ====>                                                          Scroll ===>  PAGE
    DATABASE NAME        CREATOR  BPOOL    PGSIZE PARTITIONS NTABLES SEGSIZE LO
 ** **** +-------------------------------------------------------+ **********
 __ DB2D !                                                       !       0 A
 __ DB2D !                 Select what to display                !       0 A
 ** **** !                                                       ! **********
         !                                                       !
         !                                                       !
         !                  _ partitions of tablespace           !
         !                  _ tables     in tablespace           !
         !                                                       !
         !                                                       !
         !                                                       !
         !                 Mark  _ to print output               !
         !                                                       !
         +-------------------------------------------------------+



Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +           <     >    Canc
```

A list of all main commands available on the tablespace listing screen can be invoked as a window by entering the help character "?" in the command line of the screen.

List Plans

To invoke the List Plans function, enter function code "P" on the Retrieval of System Tables screen. The name of the plan(s) to be listed must be specified. If a value followed by an asterisk is specified, all plans defined in the DB2 catalog whose names begin with this value are listed. If asterisk notation is specified only, all plans defined in the DB2 catalog are listed.



```
16:37:59              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
 PLAN *                                    S 01     Row 0 of 80 Columns 023 075
 ====>                                                Scroll ===>  PAGE
    PLAN     CREATOR     VALIDATE ISO ACQUIRE REL VALID OPER EXPLAIN     PLSIZE
 ** ******************************** top of data ********************************
 __ CAFPLAN  SAG3        R        S   U       C   Y     Y    N              2472
 __ SAGEDCL  SAG1        R        S   U       C   Y     Y    N              1992
 __ SAGESPCS SAG1        R        S   U       C   Y     Y    N              1992
 __ SAGESPRR SAG1        R        R   U       C   Y     Y    N              1992
 __ SAGTIA22 SAG1        R        S   U       C   Y     Y    N              1992
 __ SAG8BH22 SAG1        R        S   U       C   Y     Y    N              2296
 __ SAG8CC22 SAG1        R        S   U       C   Y     Y    N              4376
 __ SAG8IC22 SAG1        R        S   U       C   Y     Y    N              4264
 __ SAG8SC22 SAG1        R        S   U       C   Y     Y    N              2296
 __ SAGPLA   SAG         R        S   U       C   Y     Y    N              2648
 __ TREPH01  SAG4        B        S   U       C   A     Y    N              2168
 __ TREPLANC SAG2        R        S   U       C   N     Y    N              4560
 __ TREPLANG SAG2        R        S   U       C   N     Y    N              8976
 __ TREPLANO SAG2        R        S   U       C   N     Y    N              8976
 __ TREPLANT SAG2        R        S   U       C   Y     Y    N              2472
 __ TREPLAN1 SAG2        R        S   U       C   N     Y    N              3248

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +          <     >    Canc
```


Commands Allowed on Plans

The following line commands are available on the plan listing screen. Line commands are entered in front of the desired plan(s):

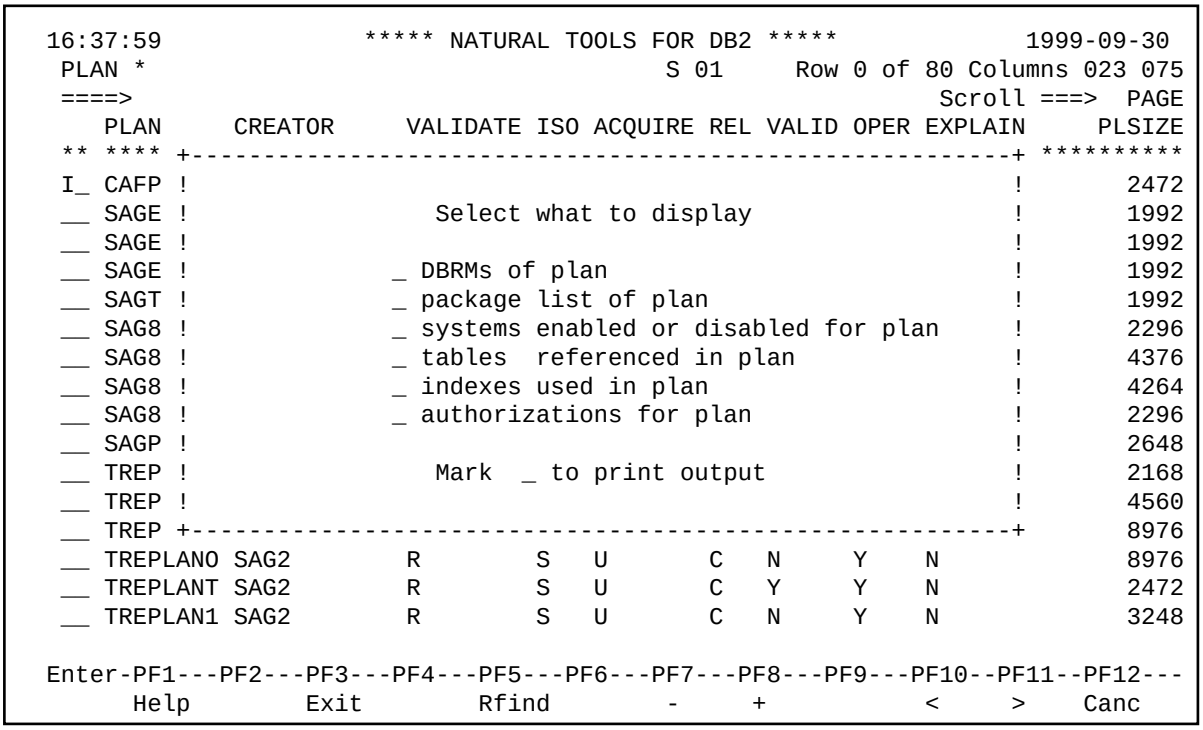
Command	Description
I	Displays information on a plan.
S	Selects a plan to be used with main commands.
U	Unselects a plan.
AU	Displays information on access rights to a plan.
DR	Displays all DBRMs contained in a plan.
IX	Displays all indexes used by a plan.
PK	Displays the package list of a plan.
SY	Displays the systems enabled or disabled for a plan.
TB	Displays tables used in a plan.

The listing displayed as a result of the "DR", "IX", "PK", or "TB" command can be used for further processing, whereas the contents of the screens displayed as a result of the "I", "AU", or "SY" command are for information purposes only.

A list of all line commands available with the List Plans function can be invoked as a window by entering the help character "?" in front of any of the listed plans.

The commands "AU", "DR", "IX", "PK", "SY", and "TB" can also be used as main commands, which are entered on the command line of the plan listing screen and apply to all plans previously selected with the "S" line command.

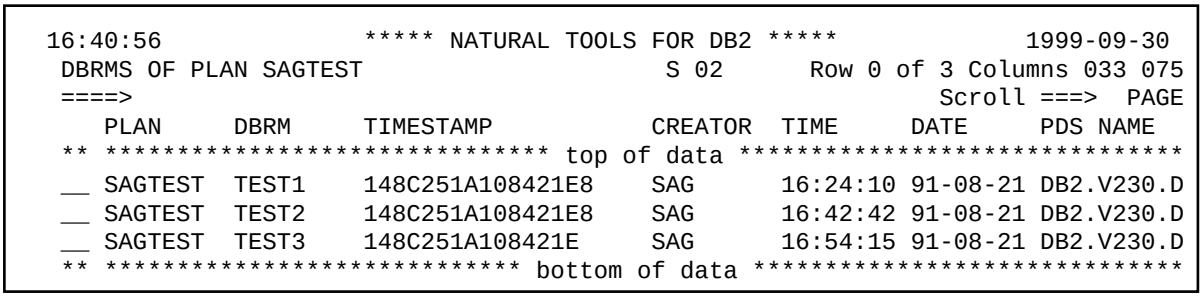
The INFO main command, which is the equivalent of the "I" line command, displays information on the DBRMs and their SQL statements contained in the plans previously selected. As with the List Database function, information resulting from the "I" or INFO commands can be printed, too.



A list of all main commands available with the List Plans function can be invoked as a window by entering the help character "?" in the command line of the plan list screen.

DBRMs of Plan

If you issue the "DR" command on the plan listing screen, a list of all DBRMs bound into the selected plan(s) is displayed.



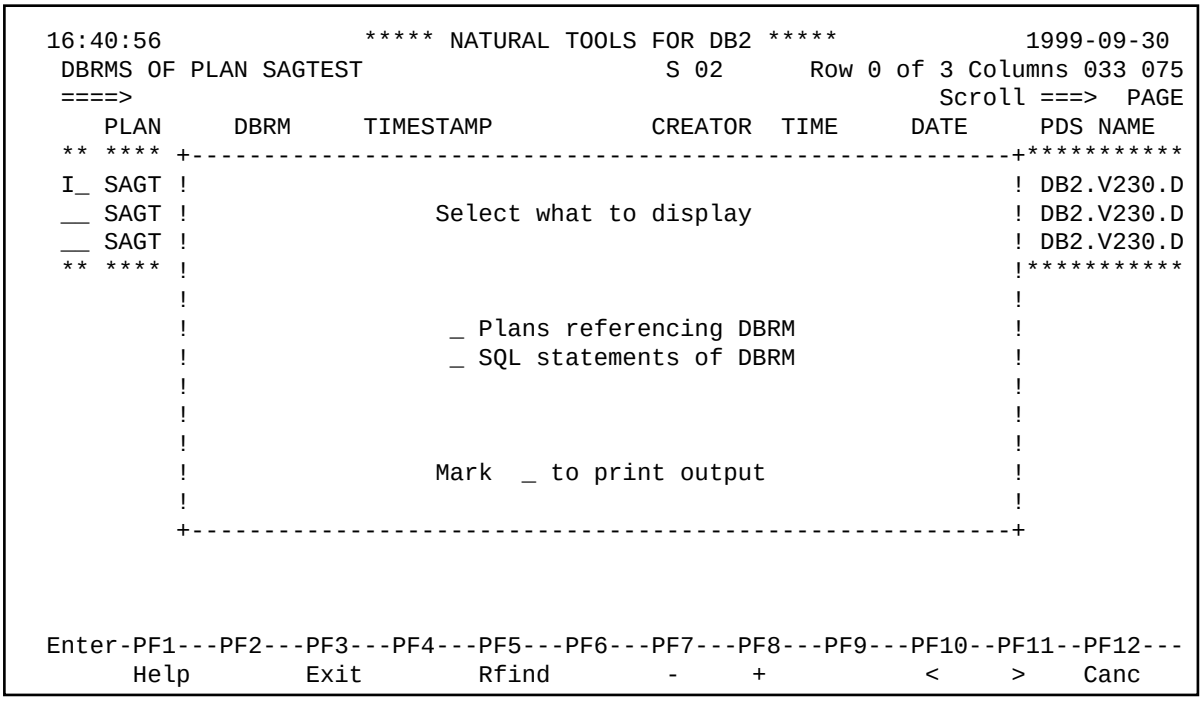
Commands Allowed on DBRMs

The following line commands are available on the DBRM listing screen. Line commands are entered in front of the desired DBRM(s):

Command	Description
I	Displays information on a DBRM.
S	Selects a DBRM to be used with main commands.
U	Unselects a DBRM.

A list of all line commands available on the DBRM listing screen can be invoked as a window by entering the help character "?" in front of any of the listed DBRMs.

The only main command that applies to DBRMs is the INFO command, which is the equivalent of the "I" line command, but displays information on all previously selected DBRMs. Instead of being displayed, all information resulting from the "I" or INFO commands can also be marked for printing. Even if already displayed, information can be printed by issuing the PRINT command.



Indexes Used in Plan

If you issue the "IX" command on either the plan listing screen or the table listing screen, a list of all indexes used in the selected plan(s) or table(s) is displayed.

```
16:40:56              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
 INDEXES OF PLAN SAGTEST                    S 02      Row 0 of 3 Columns 033 075
 ====>                                                Scroll ===>  PAGE
    CREATOR  INDEX NAME            CREATOR  TABLE NAME         COLCOUNT UNIQUE
 ** ****************************** top of data ******************************
 __ SAGCRE   XDEPT1                 SAGCRE   DEPT                      1 P
 __ SAGCRE   XEMP1                  SAGCRE   EMP                       1 P
 __ SAGCRE   XEMP2                  SAGCRE   EMP                       1 D
 ** ***************************** bottom of data *****************************
```

Commands Allowed on Indexes

The following line commands are available on the index listing screen. Line commands are entered in front of the desired index(es):

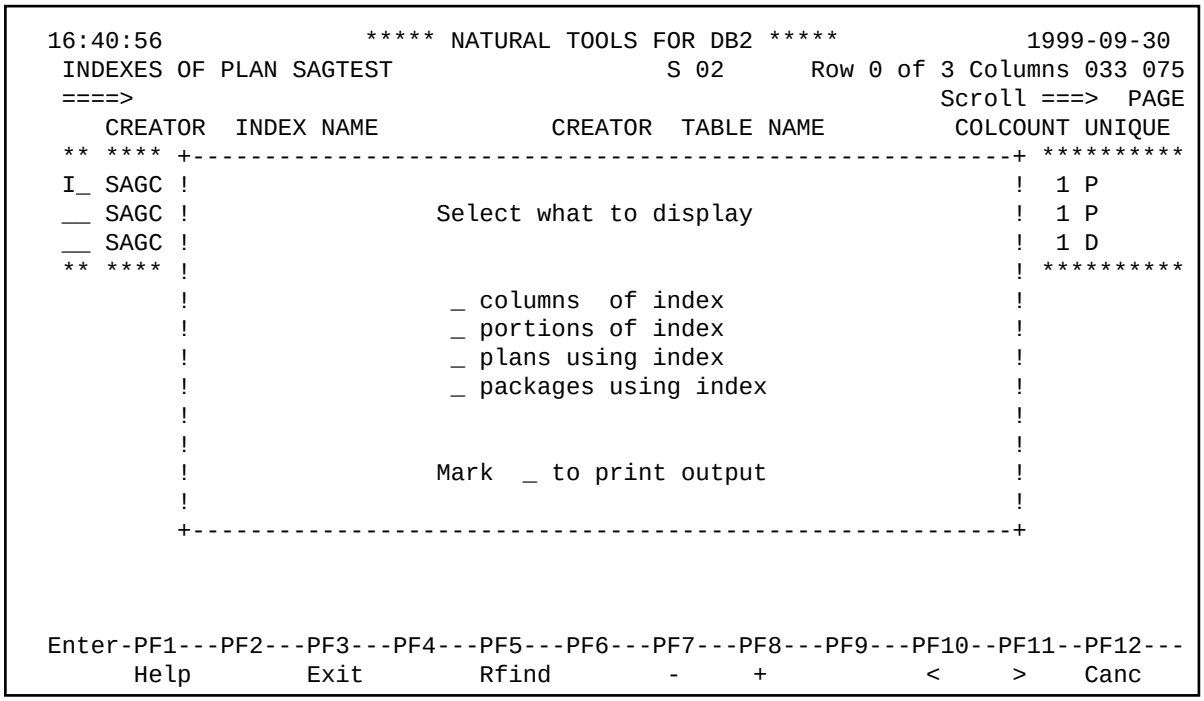
Command	Description
I	Displays information on an index.
S	Selects an index to be used with main commands.
U	Unselects an index.
CO	Displays all columns of an index.
PT	Displays the partitions of an index.

The listings of columns displayed as a result of the "CO" or "PT" command cannot be used for further processing. Like the display resulting from the "I" command, they are for information purposes only.

A list of all line commands available on the index listing screen can be invoked as a window by entering the help character "?" in front of any of the listed indexes.

The commands "CO" and "PT" can be used as main commands, too, and entered in the command line of the index listing screen. If so, all columns of all indexes previously selected with the "S" line command are displayed.

A further main command is the INFO command, which is the equivalent of the "I" line command, but displays information on all previously selected indexes. Instead of being displayed, all information resulting from the "I" or INFO commands can also be marked for printing. Even if already displayed, information can be printed by issuing the PRINT command.



A list of all main commands available on the index listing screen can be invoked as a window by entering the help character "?" in the command line of the screen.

Package List of Plan

If you issue the "PK" command on the plan listing screen, a list of all entries in the package list of the selected plan(s) is displayed.

```
16:40:56              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
 PACKAGE LIST FOR PLAN SAGTEST             S 02      Row 0 of 3 Columns 033 075
 ====>                                                 Scroll ===>  PAGE
    PLANNAME LOCATION         COLLID             NAME        SEQNO  TIMESTAMP
 ** ****************************** top of data ********************************
 __ SAGTEST                   SAGCOLLECTION      *               1  1997-09-30-
 __ SAGTEST                   SAG_STATTEST       *               2  1997-09-30-
 ** ***************************** bottom of data ******************************
```

Commands Allowed on Package List Entries

The following line commands are available on the package list screen. Line commands are entered in front of the desired package list entry:

Command	Description
I	Displays information on a package list entry.
S	Selects a package list entry to be used with main commands.
U	Unselects a package list entry.
PK	Displays all packages of a package list entry.

The listing of packages as a result of the "PK" command can be used for further processing, whereas the display resulting from the "I" command is for information purposes only.

A list of all line commands available with a package list can be invoked as a window by entering the help character "?" in front of any of the listed entries.

The command "PK" can also be used as main command, which is entered in the command line of the above screen and applies to all package list entries previously selected with the "S" line command.

List Packages

To invoke the List Packages function, enter function code "K" on the Retrieval of System Tables screen. The collection and name of the package(s) to be listed can be specified. If a value followed by an asterisk is specified, all packages defined in the DB2 catalog whose collections/names begin with this value are listed. If asterisk notation is specified only, all packages defined in the DB2 catalog are listed.

```
11:06:11              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
 PACKAGE *.*                              S 01    Row 34 of 65 Columns 041 075
 ====>                                                 Scroll ===>  PAGE
  __ COLLID             NAME        CONTOKEN CONTOKEN (HEX)   OWNER    CREATOR
  __ SAGQCATV           SAGQVPLN    ?  l?F   148C409316C67344 SAG      SAG
  __ SAGQCATV           SAGQVPPA    ?k  ? ?? 149270680F77E030 SAG      SAG
  __ SAGQCATV           SAGQVRAS    ?   ??=? 148C409B09097E28 SAG      SAG
  __ SAGQCATV           SAGQVREL    ?   ??y0 148C409C06DFA8F0 SAG      SAG
  __ SAGQCATV           SAGQVREV    ? ? ?v?  148CDFAD16A51FA0 SAG      SAG
  __ SAGQCATV           SAGQVRIL    ?  s ?B  148C40A20329C2A0 SAG      SAG
  __ SAGQCATV           SAGQVROO    ? ?  A y 148CDFAF03C18EA8 SAG      SAG
  __ SAGQCATV           SAGQVSCA    ?  u??S  148C40A409DEE2B4 SAG      SAG
  __ SAGQCATV           SAGQVSQL    ?    ??? 148C40AB001D3F30 SAG      SAG
  __ SAGQCATV           SAGQVSTM    ?   ? 7q 148C40AD078CF798 SAG      SAG
  __ SAGQCATV           SAGQVSTO    ?   ? ?  148C40B409681E68 SAG      SAG
  __ SAGQCATV           SAGQVTAB    ?   ? +U 148C40B61F024EE4 SAG      SAG
  __ SAGQCATV           SAGQVTAS    ?   ?  d 148C40B80874FF84 SAG      SAG
  __ SAGQCATV           SAGQVTBA    ?   ?  ? 148C40BB1854EC0C SAG      SAG
  __ SAGQCATV           SAGQVTBC    ?   ?d ? 148C40BD1684EC14 SAG      SAG
  __ SAGQCATV           SAGQVTBP    ?   ?    148C40BF07AE9DBC SAG      SAG
  __ SAGQCATV           SAGQVTBS    ?     ?? 148C40CA0349287C SAG      SAG

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +          <     >    Canc
```

Commands Allowed on Packages

The following line commands are available on the package listing screen. Line commands are entered in front of the desired package(s):

Command	Description
I	Displays information on a package.
S	Selects a package to be used with main commands.
U	Unselects a package.
AU	Displays information on access rights to a package.
IX	Displays all indexes used by a package.
SY	Displays all systems enabled or disabled for a package.
TB	Displays all tables used by a package.

The listings of indexes or tables displayed as a result of the "IX" or "TB" command can be used for further processing, whereas the displays resulting from the "AU", "SY", or "I" command are for information purposes only.

A list of all line commands available with the List Packages function can be invoked as a window by entering the help character "?" in front of any of the listed packages.

The commands "AU", "IX", "SY", and "TB" can also be used as main commands, which are entered in the command line of the table listing screen and apply to all tables previously selected with the "S" line command.

The INFO main command, which is the equivalent of the "I" line command, displays information on all tables previously selected. All information resulting from the "I" or INFO commands can also be printed.

```
11:06:11              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
 PACKAGE *.*                               S 01    Row 34 of 65 Columns 041 075
 ====>                                                          Scroll ===>  PAGE
 __ COLLID             NAME        CONTOKEN CONTOKEN (HEX)   OWNER    CREATOR
 i_ SAGQ +-----------------------------------------------------------+  SAG
 __ SAGQ !                                                          !  SAG
 __ SAGQ !                  Select what to display                  !  SAG
 __ SAGQ !                                                          !  SAG
 __ SAGQ !           _ systems enabled or disabled for package      !  SAG
 __ SAGQ !           _ tables  referenced in package                !  SAG
 __ SAGQ !           _ indexes used in package                      !  SAG
 __ SAGQ !           _ statements of package                        !  SAG
 __ SAGQ !           _ authorizations on package                    !  SAG
 __ SAGQ !                                                          !  SAG
 __ SAGQ !                                                          !  SAG
 __ SAGQ !                Mark  _ to print output                   !  SAG
 __ SAGQ !                                                          !  SAG
 __ SAGQ +-----------------------------------------------------------+  SAG
 __ SAGQCATV           SAGQVTBC    ?   ?d ? 148C40BD1684EC14 SAG      SAG
 __ SAGQCATV           SAGQVTBP    ?   ?    148C40BF07AE9DBC SAG      SAG
 __ SAGQCATV           SAGQVTBS    ?     ?? 148C40CA0349287C SAG      SAG

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +          <     >    Canc
```

A list of all main commands available with the List Packages function can be invoked as a window by entering the help character "?" in the command line of the packages list screen.

List Tables

To invoke the List Tables function

- Enter function code "T" on the Retrieval of System Tables screen.

The creator and name of the table(s) to be listed can be specified.

If a value followed by an asterisk is specified, all tables defined in the DB2 catalog whose creator/name begins with this value are listed.

If asterisk notation is specified only, all tables defined in the DB2 catalog are listed.

```
16:42:58              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
 TABLE SAG*.*                                 S 01   Row 34 of 361 Columns 036 075
 ====>                                                  Scroll ===>  PAGE
    CREATOR  TABLE NAME            TYPE COLCOUNT KEYCOLS RECLEN DATABASE TSNAME
 ** ****************************** top of data ******************************
 __ SAGCRE   ACT                    T           3       1     38 SAG8D22A ACT
 __ SAGCRE   DEPT                   T           4       1     59 SAG8D22A SAG8S2
 __ SAGCRE   EACT                   T           5       0     54 SAG8D22A SAG8S2
 __ SAGCRE   EDEPT                  T           6       0     75 SAG8D22A SAG8S2
 __ SAGCRE   EEMP                   T          16       0    123 SAG8D22A SAG8S2
 __ SAGCRE   EEPA                   T           8       0     52 SAG8D22A SAG8S2
 __ SAGCRE   EMP                    T          14       1    107 SAG8D22A SAG8S2
 __ SAGCRE   EMPPROJACT             T           6       0     36 SAG8D22A EMPPRO
 __ SAGCRE   EPROJ                  T          10       0     86 SAG8D22A SAG8S2
 __ SAGCRE   EPROJACT               T           7       0     45 SAG8D22A SAG8S2
 __ SAGCRE   PROJ                   T           8       1     70 SAG8D22A PROJ
 __ SAGCRE   PROJACT                T           5       3     29 SAG8D22A PROJAC
 __ SAGCRE   TCONA                  T           5       0   4056 SAG8D22P SAG8S2
 __ SAGCRE   TDSPTXT                T           3       0     91 SAG8D22P SAG8S2
 __ SAGCRE   TOPTVAL                T          11       0    354 SAG8D22P SAG8S2
 __ SAGCRE   VACT                   V           3       0      0 SAG8D22A ACT

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +           <     >    Canc
```


Commands Allowed on Tables

The following line commands are available on the table listing screen. Line commands are entered in front of the desired table(s):

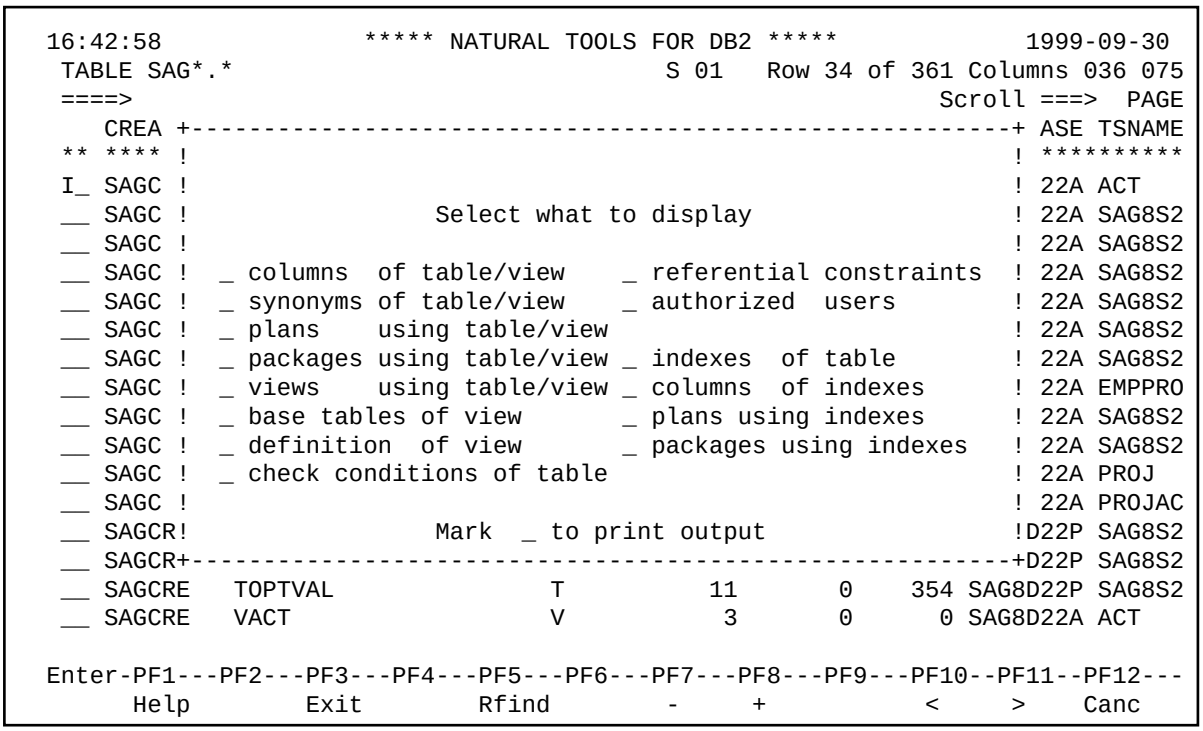
Command	Description
I	Displays information on a table.
S	Selects a table to be used with main commands.
U	Unselects a table.
AU	Displays information on access rights to a table.
CO	Displays all columns of a table.
IX	Displays all indexes on a table.
CC	Checks constraints.

The listings of indexes displayed as a result of the "IX" command can be used for further processing, whereas the listings of columns resulting from the "CO" command, as well as the displays resulting from the "AU" or "I" command, are for information purposes only.

A list of all line commands available with the List Tables function can be invoked as a window by entering the help character "?" in front of any of the listed tables.

The commands "AU", "CO", and "IX" can also be used as main commands, which are entered in the command line of the table listing screen and apply to all tables previously selected with the "S" line command.

The INFO main command, which is the equivalent of the "I" line command, displays information on all tables previously selected. All information resulting from the "I" or INFO commands can also be printed.



A list of all main commands available with the List Tables function can be invoked as a window by entering the help character "?" in the command line of the table listing screen.

User Authorizations

To invoke the User Authorization function

- Enter function code "U" on the Retrieval of System Tables screen.

The Retrieval of User Authorizations menu is displayed:

```
16:44:51              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
                      - Retrieval of User Authorizations -



                     Code Function               Parameter

                      C   Column   Authorizations Grantee
                      D   Database Authorizations Grantee
                      K   Package  Authorizations Grantee
                      P   Plan     Authorizations Grantee
                      R   Resource Authorizations Grantee
                      T   Table    Authorizations Grantee
                      U   User     Authorizations Grantee
                      ?   Help
                      .   Exit

               Code .. _   Grantee .. ________


Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                  Canc
```

The following functions are available:

Code	Description
C	Displays the columns which can be accessed by the specified grantee.
D	Displays the databases which can be accessed by the specified grantee.
K	Displays the packages which can be accessed by the specified grantee.
P	Displays the plans which can be accessed by the specified grantee.
R	Displays the resources which can be accessed by the specified grantee.
T	Displays the tables which can be accessed by the specified grantee.
U	Displays the system privileges of the specified grantee.

The following parameter must be specified:

Parameter	Description
Grantee	A list of all existing DB2 objects of the specified object type to which the specified grantee has access is displayed.

List Statistic Tables

To invoke the List Statistic Tables function

- Enter function code "S" on the Retrieval of System Tables screen.

The Retrieval of Statistic Tables menu is displayed:

```
 16:38:47              ***** NATURAL TOOLS FOR DB2 *****            1999-06-27
                         - Retrieval of Statistic Tables -


                       Code Function              Parameter

                        C   List SYSCOLSTATS      Creator, Name
                        D   List SYSCOLDISTSTATS  Creator, Name
                        I   List SYSINDEXSTATS    Index Owner, Name
                        T   List SYSTABSTATS      Creator, Name
                        ?   Help
                        .   Exit

                Code .. _   Index Owner ......... ________
                            Index Name .......... ____________________
                            Table Creator ....... ________
                            Table Name .......... ____________________



 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
       Help        Exit                                                 Canc
```

The following functions are available:

Code	Description
C	Displays the partitioned statistics for columns in a partitioned table space.
D	Displays the distribution of the values of the first column of a partitioned index.
I	Displays the statistics for a partitioned index.
T	Displays the statistics for a partitioned table space.

The following parameters must be specified:

Parameter	Description
Table Creator	The name of the creator of the table for which the statistics are to be displayed.
Table Name	The name of the table for which the statistics are to be displayed.
Index Owner	The name of the owner of the index for which the index statistics are to be displayed.
Index Name	The name of the index for which the index statistics are to be displayed.

NDB - Environment Setting

The Environment Setting facility of the Natural Tools for DB2 allows you to issue interactively special SQL statements.

For details on the SQL statements described in this section, see the relevant DB2 literature by IBM.

This section covers the following topics:

- Invoking the Environment Setting Facility
- CONNECT
- SET CURRENT SQLID
- SET CONNECTION
- SET CURRENT PACKAGESET
- SET CURRENT DEGREE
- SET CURRENT RULES
- SET CURRENT OPTIMIZATION HINT
- SET CURRENT LOCALE LC_CTYPE
- SET CURRENT PATH
- SET CURRENT PRECISION
- RELEASE

In addition, the current values of all supported Special Registers can be displayed.

- Display Special Registers

Invoking the Environment Setting Facility

To invoke the Environment Setting facility

- Entering function code "S" on the Natural Tools for DB2 Main Menu.

The Environment Setting screen is displayed, which offers you the following functions:

Environment Setting - Functions

CO	Specifies and executes the SQL statement CONNECT.
SS	Specifies and executes the SQL statement SET CURRENT SQLID.
SC	Specifies and executes the SQL statement SET CONNECTION.
SP	Specifies and executes the SQL statement SET CURRENT PACKAGESET.
SD	Specifies and executes the SQL statement SET CURRENT DEGREE.
SU	Specifies and executes the SQL statement SET CURRENT RULES.
SO	Specifies and executes the SQL statement SET CURRENT OPTIMIZATION HINT.
SL	Specifies and executes the SQL statement SET CURRENT LOCALE LC_CTYPE.
SA	Specifies and executes the SQL statement SET CURRENT PATH.
SE	Specifies and executes the SQL statement SET CURRENT PRECISION.
RE	Specifies and executes the SQL statement RELEASE.
SR	Displays the current values of the supported special registers.

CONNECT

To invoke the CONNECT function

- Enter function code "CO" on the Environment Setting screen.

The Connect screen is displayed:

```
14:23:29              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
                                  - Connect -




 >>---- CONNECT ---+-- _ ---------------------------------+-------------------><
                   !                                      !
                   !                                      !
                   +-- _ --- TO ---- _________________  --+
                   !                  (location name)     !
                   !                                      !
                   +-- _ --- RESET ----------------------+


   Current Server Version ________


Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Exec                                              Canc
```

The CONNECT statement connects the current application to a designated server. This server is the current server, which is displayed in the Current Server Version field.

On the Connect screen, you identify the current server by specifying a location name. The identified server must be known to the local DB2 subsystem.

SET CURRENT SQLID

To invoke the SET CURRENT SQLID function

- Enter function code "SS" on the Environment Setting screen.

The Set Current SQLID screen is displayed:

```
14:23:47              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
                              - Set Current SQLID -









      >>--- SET CURRENT SQLID = ----- __________ -------------------><
                                  ( USER,
                                    string-constant)









Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Exec  Free                                       Canc
```

The SET CURRENT SQLID statement changes the value of the SQL authorization identifier. With SQL statements that use unqualified table names, DB2 uses the SQLID as an implicit table qualifier. This enables you to access identical tables with the same table name but with different creator names.

On the Set Current SQLID screen, you can replace the value of CURRENT SQLID by the value of the special register USER or by a string constant. The string constant can be up to 8 characters long.

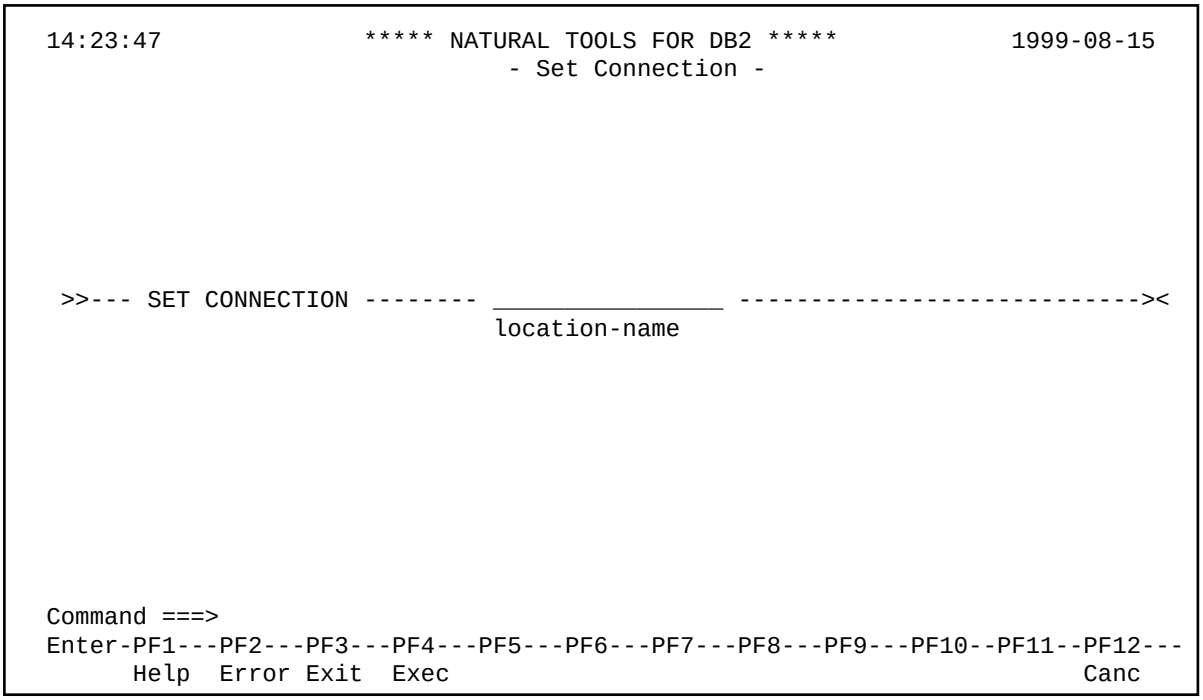
In all supported TP-monitor environments, the SQLID can then be kept across terminal I/Os until its resetting or the end of the session.

SET CONNECTION

To invoke the SET CONNECTION function

- Enter function code "SC" on the Environment Setting screen.

The Set Connection screen is displayed:



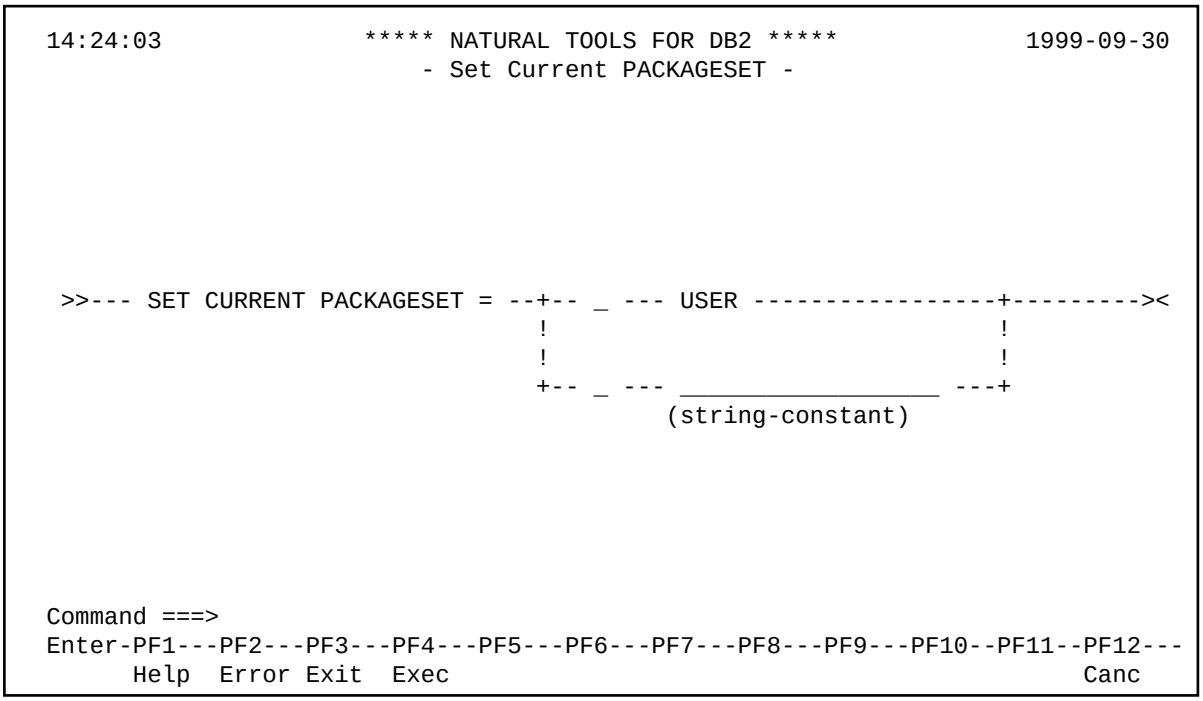
On the Set Connection screen, you identify a server by specifying a location name. The identified server must be known to the local DB2 subsystem.

SET CURRENT PACKAGESET

To invoke the SET CURRENT PACKAGESET function

- Enter function code "SP" on the Environment Setting screen.

The Set Current PACKAGESET screen is displayed:



The SET CURRENT PACKAGESET statement assigns a value to the special register CURRENT PACKAGESET.

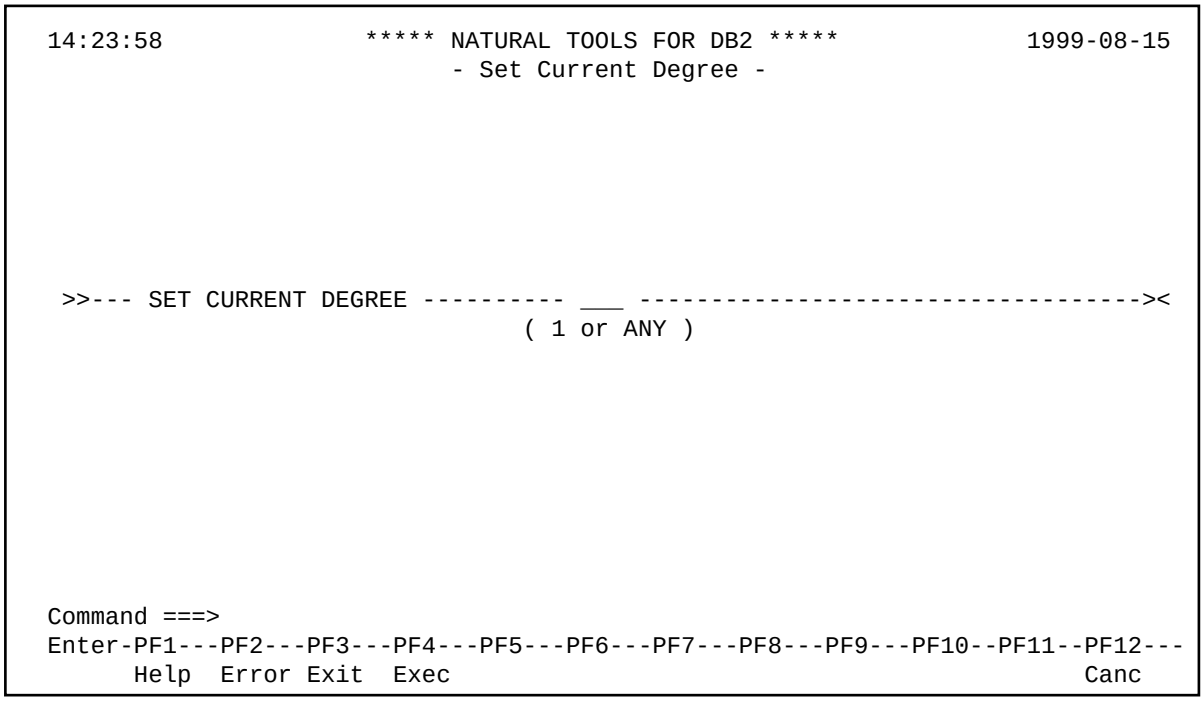
On the Set Current PACKAGESET screen, you can replace the value of CURRENT PACKAGESET by the value of the special register USER or by a string constant of up to 18 characters.

SET CURRENT DEGREE

To invoke the SET CURRENT DEGREE function

- Enter function code "SD" on the Environment Setting screen.

The Set Current Degree screen is displayed:



SET CURRENT RULES

To invoke the SET CURRENT RULES function

- Enter function code "SU" on the Environment Setting screen.

The Set Current Rules screen is displayed:


```
 14:23:58              ***** NATURAL TOOLS FOR DB2 *****            1999-08-15
                              - Set Current Rules -




  >>--- SET CURRENT RULES ----------- ___ ----------------------------------><
                                  ( DB2 or STD )









 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec                                                Canc
```

SET CURRENT OPTIMIZATION HINT

To invoke the SET CURRENT OPTIMIZATION HINT function

- Enter function code "SO" on the Environment Setting screen.

The Set Current Optimization Hint screen is displayed:

```
 14:56:55              ***** NATURAL TOOLS FOR DB2 *****            2002-08-12
                       - Set Current Optimization Hint -




>>--- SET CURRENT OPTIMIZATION HINT ---- __________ ---------------------><
                                      (string-constant)









 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec                                                Canc
```


SET CURRENT LOCALE LC_CTYPE

To invoke the SET CURRENT LOCALE LC_CTYPE function

- Enter function code "SL" on the Environment Setting screen.

The Set Current Locale LC_CType screen is displayed:

```
 14:58:12              ***** NATURAL TOOLS FOR DB2 *****             2002-08-12
                        - Set Current Locale LC_CType -









>>--- SET CURRENT LOCALE LC_CTYPE ----------------------------------------->

 >--------- _____________________________________________________ ---------><
                             (string-constant)









 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec                                          Canc
```

SET CURRENT PATH

To invoke the SET CURRENT PATH function

- Enter function code "SA" on the Environment Setting screen.

The Set Current Path screen is displayed:


```
 14:59:48              ***** NATURAL TOOLS FOR DB2 *****              2002-08-12
                             - Set Current Path -


>>- SET CURRENT PATH -----------------------------------------------------------><

   +-------------------------------<--( , )---------------------------------+
   !                                                                          !
 >-++- ____________________________________________________________________ -++-><
    !                    (schema-name<,schema-name,...>)                     !
    !                                                                        !
    +- _ ----------------------- SYSTEM PATH ----------------------------+
    !                                                                        !
    +- _ -------------------------- USER -------------------------------+
    !                                                                        !
    +- _ ---------------------- CURRENT PATH ---------------------------+
    !                                                                        !
    +- ____________________________________________________________________ -+
                   (string-constant<,string-constant,...>)

  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec                                          Canc
```

SET CURRENT PRECISION

To invoke the SET CURRENT PRECISION function

- Enter function code "SE" on the Environment Setting screen.

The Set Current Precision screen is displayed:

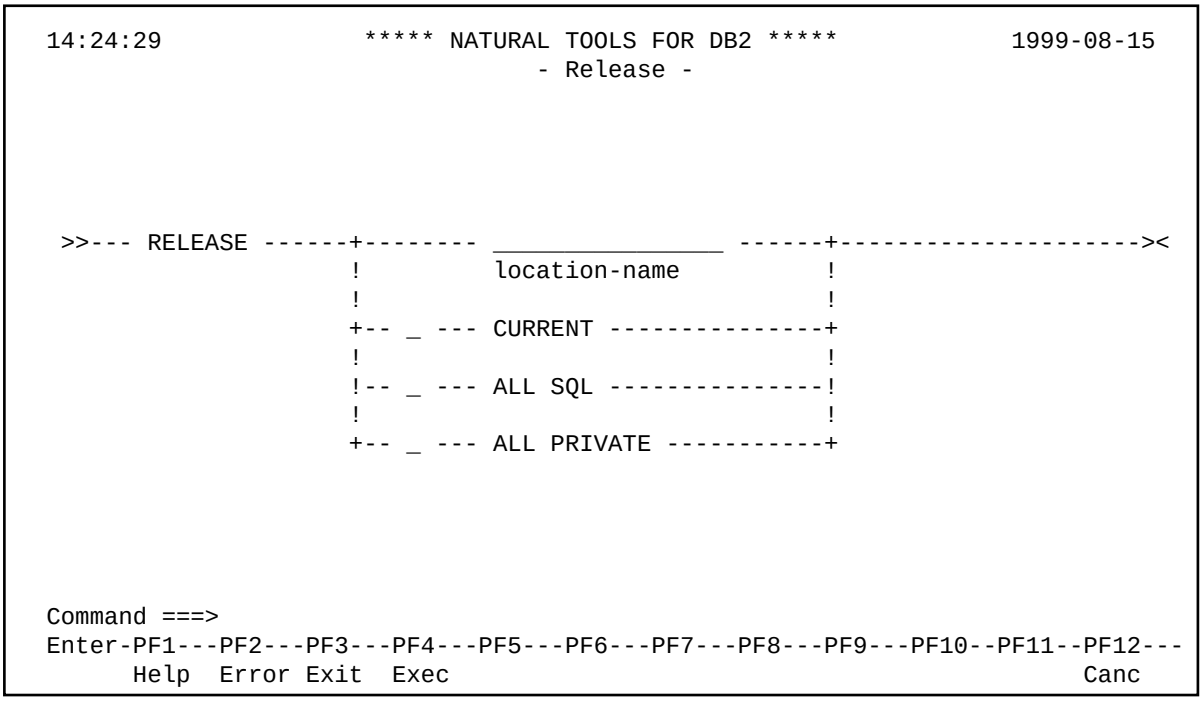
```
 15:01:17              ***** NATURAL TOOLS FOR DB2 *****              2002-08-12
                           - Set Current Precision -









>>--- SET CURRENT PRECISION -------- DEC15 ---------------------------------><
                              (DEC15,DEC31,15,31,
                          D15.1 - D15.9,D31.1 - D31.9)









  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec                                          Canc
```

RELEASE

To invoke the RELEASE function

- Enter function code "RE" on the Environment Setting screen.

The Release screen is displayed:



Display Special Registers

To invoke the Display Special Registers function

- Enter function code "SR" on the Environment Setting screen.

The Display Special Registers screen is displayed:


```
15:03:22              ***** NATURAL TOOLS FOR DB2 *****             2002-08-12
                        - Display Special Registers -


  User ............... SAG            Current Server ...... DAEFDB27
  Current SQLID ...... SAG            Current Packageset ..

  Current Degree .....                Current Rules .......
  Current OptHint ....                Current Precision ... DEC15

  Current Date ....... 12.08.2002
  Current Time ....... 15.03.22
  Current Timezone ... 20000
  Current Timestamp .. 2002-08-12-15.03.22.490284
  Current LC_CType ...
  Current Path ....... "SYSIBM","SYSFUN","SYSPROC","SAG"




 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec                                            Canc
```

The Display Special Registers screen shows you the current values of the Special Registers of DB2 supported by Natural for DB2.

NDB - Explain PLAN_TABLE

Important:
Before you use the Explain PLAN_TABLE function, refer to LISTSQL and Explain Functions in the section Special Requirements for Natural Tools for DB2.

The Explain PLAN_TABLE facility of the Natural Tools for DB2 interprets the results of SQL EXPLAIN commands from your PLAN_TABLE. The information contained in your PLAN_TABLE is represented in so-called explanations.

Explanations of a PLAN_TABLE describe the access paths chosen by DB2 to execute SQL statements.

An SQL statement is executed by DB2 in one or more steps. For each execution step, one row is inserted into the PLAN_TABLE. All rows together describing the access path for one SQL statement are called an explanation.

The explanations are identified in the PLAN_TABLE by a combination of either plan name, DBRM (database request module) name, and query number or collection name, package name, and query number.

This section covers the following topics:

- EXPLAIN Modes
- Invoking the EXPLAIN_TABLE Function
- List PLAN_TABLE - Latest Explanations
- List PLAN_TABLE - All Explanations
- Delete from PLAN_TABLE
- Explain PLAN_TABLE - Mass and Batch Processing

EXPLAIN Modes

DB2 provides three ways to explain SQL statements:

- Dynamic EXPLAIN
- Bind Plan EXPLAIN
- Bind Package EXPLAIN

Depending on the way the identifications of the explanations differ.

Dynamic EXPLAIN

Executes an SQL EXPLAIN command dynamically, where the explanation is inserted into the PLAN_TABLE of your current SQLID.

The EXPLAIN command can be issued within the Catalog Maintenance and Interactive SQL facilities of the Natural Tools for DB2. In addition, the Natural LISTSQL command can be used to extract SQL statements from cataloged Natural programs, and to issue the SQL EXPLAIN command for the extracted SQL statements.

If you issue the SQL EXPLAIN command dynamically, you should specify a query number to help identify the explanation in the PLAN_TABLE. The same query number should be used for related statements.

Depending on the method with which the DBRM used by the dynamic SQL processor is bound into the plan, DB2 uses two different ways to identify rows in the PLAN_TABLE:

- Dynamic mode
- Package mode

Dynamic Mode

The DBRM is bound directly into the plan.

When an explanation is inserted, the plan name, the DBRM name, and the query number are determined by DB2 as follows:

plan name	is left blank;
DBRM name	is the name of the DBRM used by the dynamic SQL processor;
query number	is equal to the query number you specified with the EXPLAIN command (the default query number is "1").

This explanation mode is called dynamic mode.

Package Mode

The DBRM is bound as package into the plan.

When an explanation is inserted, the collection name, the package name, and the query number are determined by DB2 as follows:

collection name	is the name of the collection that contains the package;
package name	is the name of the package used by the dynamic SQL processor;
query number	is equal to the query number you specified with the EXPLAIN command (the default query number is "1").

This explanation mode is called package mode.

Bind Plan EXPLAIN

Binds an application plan with the option EXPLAIN YES, where the explanation is inserted into the PLAN_TABLE of the owner of the plan. When an explanation is inserted, the plan name, the DBRM name, and the query number are determined by DB2 as follows:

plan name	is the name of the plan;
DBRM name	is the name of the DBRM that contains the SQL statement;
query number	is equal to the statement number (*stmtno*), which is generated by the DB2 precompiler.

Bind Package EXPLAIN

Binds a package with the option EXPLAIN YES, where the explanation is inserted into the PLAN_TABLE of the owner of the package. When an explanation is inserted, the collection name, the package name, and the query number are determined by DB2 as follows:

collection name	is the name of the collection that contains the package;
package name	is the name of the package that contains the SQL statement;
query number	is equal to the statement number (*stmtno*), which is generated by the DB2 precompiler.

Invoking the EXPLAIN_TABLE Function

Explanations can be selected by either plan name, DBRM name, and query number or collection name, package name, and query number. If you issue an EXPLAIN command various times, it is possible that multiple explanations are identified by a given combination of these selection fields. Thus, you can select either all explanations or only the most recent one. A list with all selected explanations is displayed, from which you can select individual rows for a more detailed description.

The individual rows of a PLAN_TABLE are displayed one row per line. Rows that describe the same SQL statement are shown together as one explanation. Different explanations, are separated by empty lines. You can browse through the list and select a detailed report for individual explanations. If rows have been inserted into your PLAN_TABLE as a result of a Natural LISTSQL command, the names of the Natural library and program are also displayed.

To invoke the Explain PLAN_TABLE facility

- Enter function code "X" on the Natural Tools for DB2 Main Menu.

The Explain PLAN_TABLE screen is displayed:

```
16:45:35              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
                            - Explain PLAN_TABLE -


                    Code Function

                     L   List PLAN_TABLE - Latest Explanations
                     A   List PLAN_TABLE - All    Explanations
                     D   Delete from PLAN_TABLE
                     ?   Help
                     .   Exit

               Code .. _   Mode ........ DYNAMIC_ ( Dynamic, Plan, Package )
                           Plan ........ _________
                           Collection .. _________
                           DBRM/Package  _________
                           Queryno ..... _____ - _____



Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                   Canc
```

The following functions are available:

Code	Description
L	The List PLAN_TABLE - Latest Explanations function lists the last explanation for any combination of the parameters described below.
A	The List PLAN_TABLE - All Explanations function lists all explanations for any combination of the parameters described below.
D	The Delete from PLAN_TABLE function deletes the specified explanations from your PLAN_TABLE.

The following parameters can be specified:

Parameter	Description
Mode	Specifies the explanation mode (Dynamic, Plan, or Package).
Plan *plan-name*	Specifies a valid plan name. The parameter Plan is only required in Plan mode.
Collection *collection-name*	Specifies a valid collection name. The parameter Collection is only required in Package mode.
DBRM/Package dbrm/package-name	In Plan mode, specifies a valid DBRM name. In Package mode,specifies a valid package name. In dynamic mode, specifies the DBRM used by the dynamic SQL processor. If a value followed by an asterisk (*) is specified, all DBRMs/packages of the specified plan/collection whose names start with the specified value are considered. If asterisk notation is specified only, all DBRMs/packages of the specified plan/collection are considered. The DBRM/Package parameter is used to limit the display to individual DBRMs/packages.
Queryno *no.1 - no.2*	This parameter specifies a valid range of query numbers, where the following rules apply: If no query number is specified, all query numbers are displayed; If only the first query number is specified, only this query number is displayed; If only the second query number is specified, all query numbers up to and including the second query number are displayed; If both query numbers are specified, all query numbers between and including the first and the second query number are displayed.

List PLAN_TABLE - Latest Explanations

This function only lists the most recent explanation for any specified combination of either plan name, DBRM name, and query number or package name, collection name and query number.

List PLAN_TABLE - All Explanations

This function lists all explanations for any combination of either plan name, DBRM name, and query number or package name, collection name and query number. The query number parameters are interpreted as above.

Sample Listing of Explanations

```
16:48:24              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
 Plan TESTPLAN                              S 01     Row 0 of 82 Columns 048 100
 ====>                                                 Scroll ===>  PAGE
    DBRM     QNO   ME ACC MA IO     PRE SORTN SORTC TCREATOR TABLENAME
 ** ******************* top of data ****************************************
 __ TEST     722      I   1  -           ----  ---- SAGCRE   DEPT
 __ TEST     722   1  I   1  -           ----  ---- SAGCRE   EMP
 __ TEST     722   3        -           ----  --O-
 __ TEST     722      I   1  -           ----  ---- SAGCRE   DEPT
 __ TEST     722      I   1  Y           ----  ---- SAGCRE   EMP
 __ TEST     722      I   1  -           ----  ---- SAGCRE   DEPT
 __
 __ TEST     761      I   1  -           ----  ---- SAGCRE   EMP
 __ TEST     761   1  I   1  -           ----  ---- SAGCRE   DEPT
 __ TEST     761   3        -           ----  --O-
 __ TEST     761      I   1  -           ----  ---- SAGCRE   EMP
 __ TEST     761      I   1  Y           ----  ---- SAGCRE   DEPT
 __
 __ TEST     793      I   1  -           ----  ---- SAGCRE   DEPT
 __ TEST     793   1  I   1  -           ----  ---- SAGCRE   EMP
 __ TEST     793   1  I   1  -           ----  ---- SAGCRE   EMP

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +          <     >    Canc
```

Commands Available

The following line commands are available within listings of the Explain PLAN_TABLE facility. Line commands are entered in front of any of the rows of the desired explanation(s).

Command	Description
I	Displays a window where additional information about an explanation can be selected
S	Selects an explanation to be used with the INFO command described below.
U	Unselects an explanation for use with the INFO command.

A list of the line commands available can be invoked as a window by entering the help character "?" in front of any of the listed rows.

Apart from the line commands, the INFO command can be specified, too. The INFO command must be entered in the command line of the listing screen and is the equivalent of the "I" line command. INFO displays a window where additional information can be selected on all explanations previously selected by the "S" line command.

In Plan mode, the following window is displayed, where you can select which additional information you want to be displayed or printed.

```
16:48:24              ***** NATURAL TOOLS FOR DB2 *****              1999-09-30
 Plan TESTPLAN                              S 01     Row 0 of 82 Columns 048 100
 ====>                                                      Scroll ===>  PAGE
    DBRM     QNO   ME ACC MA IO     PRE SORTN SORTC TCREATOR TABLENAME
 ** **** +-----------------------------------------------------+***********
 __ TEST !                                                     !
 __ TEST !                 Select what to display              !
 __ TEST !                                                     !
 __ TEST !                 _ information about plan            !
 __ TEST !                 _ statements of plan                !
 __ TEST !                 _ data from PLAN_TABLE              !
 __      !                 _ evaluation of PLAN_TABLE          !
 __ TEST !                 _ catalog statistics                !
 __ TEST !                 _ columns of used indexes           !
 __ TEST !                                                     !
 __ TEST !                 Mark  _ to print output             !
 __ TEST !                                                     !
 __      +-----------------------------------------------------+
 __ TEST     793      I   1  -           ----  ---- SAGCRE   DEPT
 __ TEST     793   1  I   1  -           ----  ---- SAGCRE   EMP
 __ TEST     793   1  I   1  -           ----  ---- SAGCRE   EMP

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +           <     >    Canc
```

Accordingly, the following window is displayed in Package mode:

```
16:48:24              ***** NATURAL TOOLS FOR DB2 *****              1999-09-30
 Package TESTPACK                           S 01     Row 0 of 82 Columns 048 100
 ====>                                                      Scroll ===>  PAGE
    DBRM +-----------------------------------------------------+
 ** **** !                                                     ! **********
 __ TEST !                                                     ! ES
 __ TEST !                 Select what to display              ! ES
 __ TEST !                                                     ! ES
 __ TEST !                 _ information about package         ! ES
 __ TEST !                 _ statements of package             ! ES
 __ TEST !                 _ data from PLAN_TABLE              ! ES
 __      !                 _ evaluation of PLAN_TABLE          ! ES
 __ TEST !                 _ catalog statistics                ! ES
 __ TEST !                 _ columns of used indexes           ! ES
 __ TEST !                                                     ! ES
 __ TEST !                 Mark  _ to print output             ! ES
 ** **** +-----------------------------------------------------+ **********




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +           <     >    Canc
```

Browsing of data displayed is performed with browse commands, of which the most important can also be issued via PF keys.

Option	Description
Information on Plan/Package	If a plan/package name has been specified, this option includes information from the DB2 catalog, such as date and time of the bind, as well as several bind options. In Dynamic mode, this option is not available.
Statements of Plan/Package	If a plan/package name has been specified, this option provides information on the explained SQL statements contained in this package. This information is taken from the DB2 catalog. In Dynamic mode, this option is not available.
Data from PLAN_TABLE	This option provides information from the PLAN_TABLE about the selected rows.
Evaluation of PLAN_TABLE	This option provides a description of the PLAN_TABLE. For each execution step, it describes: the locks chosen by DB2, whether a join operation is performed, whether the data is sorted and why the sort is performed, the access path in detail.
Catalog Statistics	This option provides statistical information from the DB2 catalog.
Columns of Indexes	This option provides the columns of used indexes including catalog statistics on this columns.

Delete from PLAN_TABLE

The Delete from PLAN_TABLE function is also used to select PLAN_TABLE explanations depending on the specified combination of either plan name, DBRM name, and query number or collection name, package name, and query number. This time, however, the selected PLAN_TABLE explanations are not displayed but deleted.

The Delete from PLAN_TABLE function is useful to delete old data before either binding or rebinding a plan, or before executing an SQL EXPLAIN command.

To prevent PLAN_TABLE explanations from being deleted unintentionally, you are prompted for confirmation:

```
16:50:23              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
                          - Delete from PLAN_TABLE -



        The SQL Command

             DELETE FROM PLAN_TABLE
                WHERE APPLNAME = ' '
                  AND COLLID = 'OLD'
                  AND PROGNAME LIKE 'ANY%'
                  AND QUERYNO BETWEEN 1 AND 2

        will be executed.



        Press PF5 to delete the data from the PLAN_TABLE or
              PF3 to return to the menu without deleting data

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Del                                       Canc
```

Apart from the global PF-key settings, with the Delete from PLAN_TABLE function of the Explain PLAN_TABLE facility, PF5 (Del) is used to confirm the deletion of previously selected explanations.

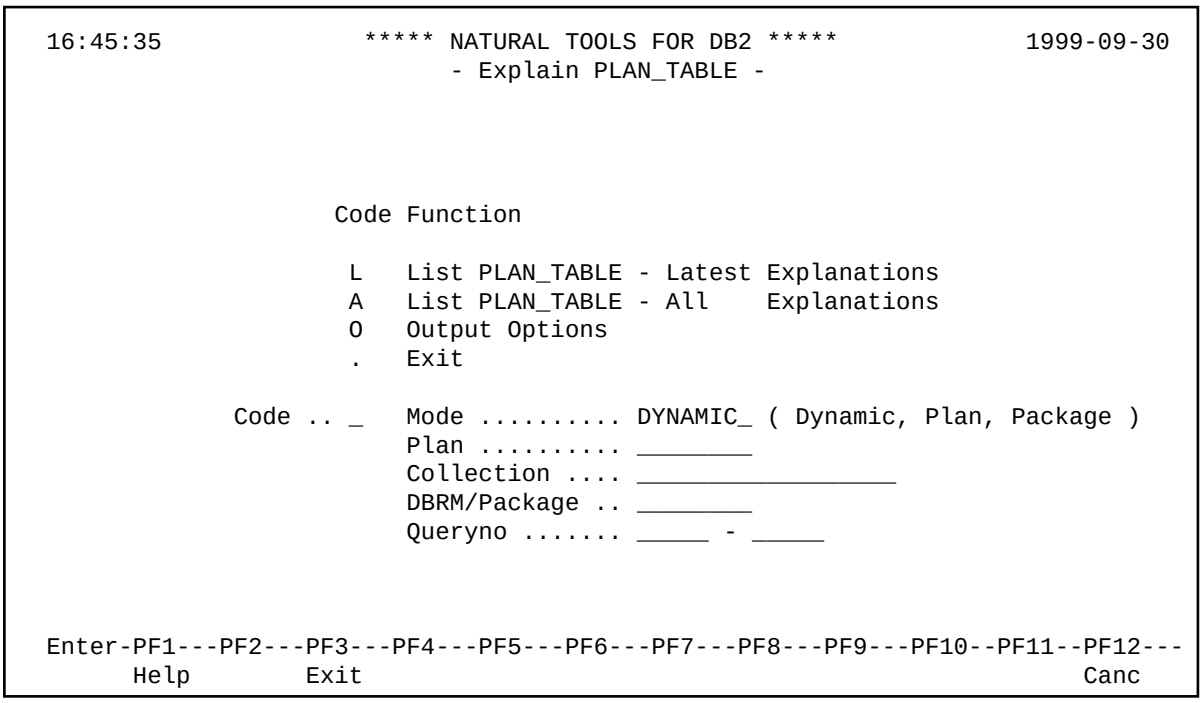
Explain PLAN_TABLE Facility for Mass and Batch Processing

An adapted version of the Explain PLAN_TABLE facility is also available for online mass processing and for batch mode execution.

EXPLAINB for Mass Processing

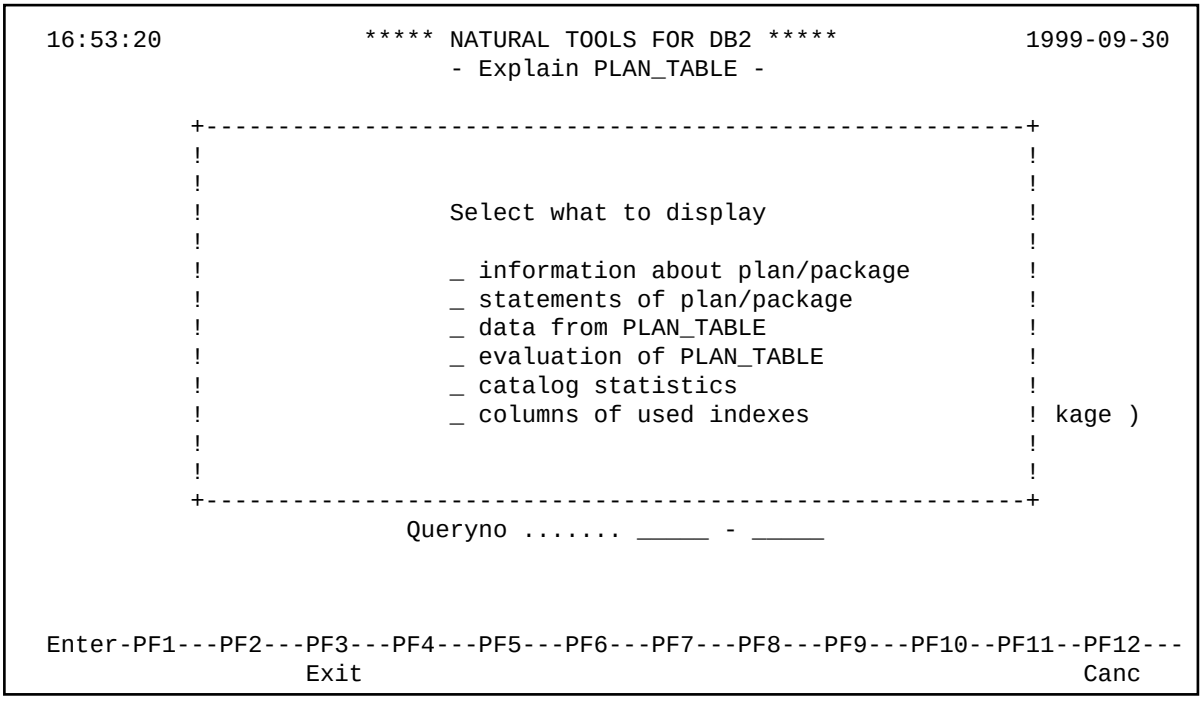
For online mass processing, a modified version of the Explain PLAN_TABLE facility is available.

To invoke this facility, LOGON to the Natural system library SYSDB2 and enter the command EXPLAINB. The following screen is displayed:



In addition to function codes "L" (List PLAN_TABLE - Latest Entries function) and "A" (List PLAN_TABLE - All Entries function), function code "O" (Output Options) is available.

The Output Options function enables you to restrict the output of information on PLAN_TABLE entries. The various options are listed in a window invoked by entering function code "O" on the above Explain PLAN_TABLE menu. The window is similar to the one invoked by the online "I" or INFO commands.



If the Output Options function has been selected, only information covered by the options marked for output are printed.

If function code "O" has not been selected, all information on PLAN_TABLE entries covered by the options listed in the above window are printed.

In both cases, you are prompted for a printer.

EXPLAINB in Batch Mode

Apart from being used for online mass processing, the functionality of EXPLAINB is especially intended for batch processing. If EXPLAINB is used in batch mode, output is sent to a dataset referred to by DD name CMPRT01 (logical printer 1).

NDB - File Server Statistics

If a file server has been installed, the file server statistics part of the Natural Tools for DB2 is used to display statistics on the use of the file server.

To invoke the File Server Statistics function

- Enter function code "F" on the Natural Tools for DB2 Main Menu.

The File Server - Generation Statistics screen is displayed:

```
16:53:20              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
                    - File Server - Generation Statistics -


  File Server Dataset Name ......: SAG.N2122.FSERV

  Enqueue Resource Name .........: FSERVV609

  Total Number of File Server Blocks ..........: 1000

  File Server Block Size ......................: 4080

  Number of Space Map Blocks ..................: 2

  Number of Global Directory Blocks ...........: 1
                             Entries ..........: 203
  User Space Allocation Quantities Primary ....: 50
                                   Secondary ..: 10
  Total Number of Blocks permitted per User ...: 200

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                          Next  Canc
```

This screen provides information on parameters that must be specified when generating the file server.

If the file server storage medium is the Software AG Editor buffer pool, the File Server - Generation Statistics screen looks as follows:

```
16:53:20              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
                    - File Server - Generation Statistics -


  File Server Dataset Name ......: STORAGE MEDIUM IS EDITOR BUFFER POOL

  Enqueue Resource Name .........:

  Total Number of File Server Blocks ..........: 0

  File Server Block Size ......................: 4088

  Number of Space Map Blocks ..................: 0

  Number of Global Directory Blocks ...........: 0
                             Entries ..........: 0
  User Space Allocation Quantities Primary ....: 20
                                   Secondary ..: 10
  Total Number of Blocks permitted per User ...: 100

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                           Next  Canc
```

If you press PF11 (Next), a second screen is displayed, the File Server - User Statistics screen, showing statistics that have been kept since the file server was installed - Statistics since Generation -, and statistics about the current Natural session - Current Session Statistics.

```
16:53:20              ***** NATURAL TOOLS FOR DB2 *****            1999-09-30
                       - File Server - User Statistics -

    Statistics since Generation:

   Active Users - Maximum Number: 3          Current Number: 1
   Maximum Number of used Blocks for single User ..........: 200
                                for all Users .............: 200
   Number of Block Allocations PRIMARY .....................: 13
                               SECONDARY ...................: 17
   Number of free Blocks ...................................: 997
   Number of INIT SESSION Calls ............................: 65

    Current Session Statistics:

   Total Number of Blocks ................. ...: 0
                  Free Blocks .................: 0
                  Secondary  Allocations ......: 0
   VSAM I/O Buffer inside DB2AREA ........... ..: YES   (Yes/No)

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                     Prev        Canc
```


If you press PF10 (Prev), you are returned to the File Server - Generation Statistics screen.

Statistics are updated, each time you press ENTER, PF10, or PF11.

If the file server storage medium is the Software AG Editor buffer pool, the user Statistics screen looks as follows:

```
16:53:20              ***** NATURAL TOOLS FOR DB2 *****             1999-09-30
                       - File Server - User Statistics -

   Statistics since Generation:

   Active Users - Maximum Number: 3          Current Number: 0
   Maximum Number of used Blocks for single User ..........: 0
                                 for all Users ............: 0
   Number of Block Allocations PRIMARY ....................: 0
                               SECONDARY ..................: 0
   Number of free Blocks ..................................: 0
   Number of INIT SESSION Calls ...........................: 0

   Current Session Statistics:

   Total Number of Blocks ................. ...: 20
                   Free Blocks .................: 20
                   Secondary  Allocations ......: 0
   VSAM I/O Buffer inside DB2AREA ........... ..: YES   (Yes/No)

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                      Prev        Canc
```

Note that the section "Statistics since Generation" could not be provided by this display.

Issuing DB2 Commands from Natural

The DB2 Command part of the Natural Tools for DB2 enables you to issue DB2 commands from a Natural environment.

A file is maintained for each user on the FUSER file. This file is stored under the object name DB2$CMD in the Natural library of the current user.

You can select a command and submit it, save the command file and save and/or print the output report.

This section covers the following topics:

- Invoking the DB2 Command Part
- Displaying the Command File
- Displaying the Output Record

Invoking the DB2 Command Part

To invoke the Interactive SQL function

- Enter function code "D" on the Natural Tools for DB2 Main Menu.

The Execute DB2 Command screen is displayed:

```
    16:07:56              ***** NATURAL TOOLS FOR DB2 *****            1999-07-02
                               - Execute DB2 Command -




                           Code   Function

                             C    Display Commands
                             O    Display Output
                             ?    Help
                             .    Exit

                     Code .. _    Library .. DBA_____




   Command ===>
   Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
         Help        Exit                                                  Canc
```


The following functions are available:

Code	Decription
C	Displays your command file. If you have not saved a command file yet, a default file is displayed.
O	If an output file exists, the output report is displayed.

The following parameter can be specified:

Library: You can enter a user name or library. The default is the currrent user ID.

Displaying the Command File

To display the command file

- Enter function code "C" on the DB2 command menu.

The DB2 Commands screen is displayed:

```
16:12:11              ***** NATURAL TOOLS FOR DB2 *****           1999-07-02
                                - DB2 Commands -

  Mark the line of the command you want to execute with 'S' and press PF4

  Cmd 1    _    -DISPLAY THREAD (*).....................................
  Cmd 2    _    -DISPLAY LOCATION.......................................
  Cmd 3    _    -DISPLAY DATABASE(*) LIMIT(2500)........................
                ........................................................
  Cmd 4    _    -DISPLAY PROCEDURE (*)..................................
                ........................................................
  Cmd 5    _    -DISPLAY DATABASE(DSNDB04) LIMIT (*)....................
                ........................................................
  Cmd 6    _    ........................................................
                ........................................................
  Cmd 7    _    ........................................................
                ........................................................
  Cmd 8    _    ........................................................
                ........................................................
                ........................................................

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit  Subm  Save                                Next  Canc
```

Use PF11 to scroll to the next page.

You can modify the command file. Save your modifications with PF5.

To execute a command

- Mark the command with "S" and press PF4.

The results are displayed on the DB2 Commands Output screen:

```
16:13:23              ***** NATURAL TOOLS FOR DB2 *****          1999-07-02
                            - DB2 Commands Output -

  Command:           -DISPLAY DATABASE(DSNDB04) LIMIT (*)
  Return Code 1:     00000000          Return Code 2:   00000000
  Length of Output:  00001AFB

  DSNT360I - **********************************
  DSNT361I - *  DISPLAY DATABASE SUMMARY
             *    GLOBAL
  DSNT360I - **********************************
  DSNT362I -     DATABASE = DSNDB04  STATUS = RW
                 DBD LENGTH = 72674
  DSNT397I -
  NAME     TYPE PART STATUS             PHYERRLO PHYERRHI CATALOG  PIECE
  -------- ---- ---- ------------------ -------- -------- -------- -----
  ADRESSE  TS        RW
  ALIASRBY TS        RW
  ALLDATA0 TS        RW


Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit        Save  --    -     +     ++                Canc
```

To save the command file, use PF5; the output file is stored under the object name DB2$OUT in the Natural library of the current user.

To return to the command file, use PF3.

You can submit further commands.

Displaying the Output Report

To display the last output record

- Enter function code "O" on the DB2 command menu.

The DB2 Commands Output screen is displayed:

```
16:13:57              ***** NATURAL TOOLS FOR DB2 *****           1999-07-02
                             - DB2 Commands Output -

  Command:           -DISPLAY DATABASE(*) LIMIT(2500)
  Return Code 1:     00000000          Return Code 2:   00000000
  Length of Output:  00007468

  DSNT360I - **********************************
  DSNT361I - *  DISPLAY DATABASE SUMMARY       *
             *    GLOBAL                       *
  DSNT360I - **********************************
  DSNT362I -     DATABASE = DSNDB01  STATUS = RW
                 DBD LENGTH = 8000
  DSNT397I -
  NAME     TYPE PART STATUS             PHYERRLO PHYERRHI CATALOG  PIECE
  -------- ---- ---- ------------------ -------- -------- -------- -----
  DBD01    TS        RW
  SPT01    TS        RW
  SCT02    TS        RW


Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit        Print --    -     +     ++                Canc
```

To print the output record, use PF5.

Natural System Commands for DB2

The following system commands have been incorporated into the Natural Tools for DB2:

- LISTSQL Command
  lists Natural DML statements and their corresponding SQL statements.
- LISTSQLB Command
  provides explanations of SQL statements for a specific object.
- SQLERR Command
  provides diagnostic information on a DB2 error.
- LISTDBRM Command
  displays either a list of DBRMs (database request modules) for a particular Natural program or a list of Natural programs that reference a particular DBRM.

LISTSQL Command

Important:
Before you use the LISTSQL command, refer to LISTSQL and Explain Functions in the section Special Requirements for Natural Tools for DB2.

LISTSQL [*object-name* **]**

The LISTSQL command lists the Natural statements in the source code of a programming object that are associated with a database access, and the corresponding SQL statements into which they have been translated. LISTSQL is issued from the Natural NEXT prompt.

Thus, before executing a Natural program which accesses a DB2 table, you can view the generated SQL code by using the command LISTSQL.

If a valid object name is specified, the object to be displayed must be stored in the library to which you are currently logged on.

If no object name is specified, LISTSQL refers to the object currently in the Natural source area.

The generated SQL statements contained in the specified object are listed one per page.

Sample LISTSQL Screen

```
17:01:25              ***** NATURAL TOOLS FOR SQL *****              1999-09-30
 Member RTTB--IN                  - LISTSQL -                  Library TEST

 NATURAL statement at line 0910                                    Stmt   1 / 7

   FIND SYSCOLUMNS WITH TBCREATOR = #TBCREATOR AND TBNAME = #TBNAME
   SORTED BY COLNO


 generated SQL statement   Mode : dynamic  DBRM :                  Line   1 / 5

   SELECT  NAME, COLNO, COLTYPE, LENGTH, SCALE, NULLS, DEFAULT, KEYSEQ
   FROM    SYSSAG.SYSCOLUMNS
   WHERE   TBCREATOR = ? AND TBNAME = ?
   ORDER BY COLNO
   FOR FETCH ONLY




Command ===>                                       Queryno for EXPLAIN 1____
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
           Error Exit  Expl        Parms  -     +          Prev  Next  Canc
```

Within the listed results, you can go from one listed SQL statement to another by pressing PF10 (Prev) or PF11 (Next). If a single SQL statement does not fit on the screen, you can scroll backwards or forwards by pressing PF7 or PF8, respectively.

If a static DBRM has been generated, the name of this DBRM is displayed in the DBRM field of the LISTSQL screen; otherwise, the DBRM field remains empty.

If an error occurs, PF2 (Error), which executes the SQLERR command, can be used to provide information on DB2 errors.

With PF4 (Expl), a DB2 EXPLAIN command can be executed for the SQL statement currently listed. The query number (Queryno) for the EXPLAIN command is set to "1" by default, but you can overwrite this default.

With PF6 (Parms), a further screen is displayed which lists all parameters from the SQLDA for the currently displayed SQL statement:

```
17:01:27              ***** NATURAL TOOLS FOR SQL *****              1999-09-30
 Member RTTB--IN                   - LISTSQL -                  Library TEST

          Mode : dynamic   DBRM :           Contoken :

          static parms :

          SQLDA

       Nr  Type   Length
        1. CHAR       18             0728 0000 0012 01C5 0000 0000 0601 0000
        2. SMALLINT    2             073A 0000 0002 01F5 0000 0000 0201 0000
        3. CHAR        8             073C 0000 0008 01C5 0000 0000 0201 0000
        4. SMALLINT    2             0744 0000 0002 01F5 0000 0000 0201 0000
        5. SMALLINT    2             0746 0000 0002 01F5 0000 0000 0201 0000
        6. CHAR        1             0748 0000 0001 01C5 0000 0000 0201 0000
        7. CHAR        1             0749 0000 0001 01C5 0000 0000 0201 0000
        8. SMALLINT    2             074A 0000 0002 01F5 0000 0000 0201 0000

        1. CHAR        8             0290 0108 0008 01C4 0000 0000 0000 0000

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                 Exit                     -     +                      Canc
```

In static mode, static information is also displayed, which includes the static DBRM name, the DB2 consistency token and some internal static parameters.

DB2 EXPLAIN Command

Important:
Before you use the EXPLAIN command, refer to LISTSQL and Explain Functions in the section Special Requirements for Natural Tools for DB2.

The EXPLAIN command provides information on the DB2 optimizer's choice of strategy for executing SQL statements. For the EXPLAIN command to be executed, a PLAN_TABLE must exist. The information determined by the DB2 optimizer is

to this table. The corresponding explanation is read from the PLAN_TABLE and displayed via the EXPLAIN Result screen.

Sample Explain Result Screen

```
17:02:29              ***** NATURAL TOOLS FOR SQL *****             1999-09-30
 Queryno 1                    - EXPLAIN Result -                    Row  1 / 2

                       Estimated cost :   107.1  timerons

 Qblockno Mixop   Access Match Index  Pre-         Access-              Column-
    Planno seq     type   cols  only fetch   creator      name          fn_eval
  ---  --- ---     ----  ----- ----- -----   -------- ------------------   -
  1    1            I      2                SYSIBM   SAGDCX01
  1    2



       Table                          Tslock            -- sortn --   -- sortc --
 Tabno Creator      Tname              mode      Method uq jo or gr   uq jo or gr
  ---  -------- ------------------  ------     ------ -- -- -- --   -- -- -- --
  1    SYSIBM   SYSCOLUMNS             IS              N  N  N  N    N  N  N  N
                                                  3    N  N  N  N    N  N  Y  N



Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                 Exit  Info               -     +                      Canc
```

If an explanation does not fit on one screen, you can scroll backwards and forwards by pressing PF7 or PF8, respectively.

The value in the Estimated cost field is taken from SQLERRD (4) in the SQLCA; it is a rough estimate of the required resources.

With PF4 (Info), the additional information that is provided with the EXPLAINB command is displayed.

LISTSQLB Command

Important:
Before you use the LISTSQLB command, refer to LISTSQL and Explain Functions in the section Special Requirements for Natural Tools for DB2.

The command LISTSQLB can be executed in batch mode or issued online from the Natural NEXT line.

If executed online, the following screen is invoked:

```
16:54:35              ***** NATURAL Tools for SQL *****           1999-09-30
                                 - LISTSQLB -




                          Code Function

                           X   Explain all SQL Statements
                           .   Exit

                    Code .. _   Member ... ________
                                Queryno .. 1____




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit                                                  Canc
```

By specifying a valid member name, the explanation of SQL statements can be limited to certain member(s); an asterisk (*) can be used for range specification:

- If you specify a unique member name, all SQL statements contained in this member are explained;
- If you specify a value followed by an asterisk, all SQL statements contained in all members with names beginning with the specified value are explained;
- If you specify an asterisk only (or leave the field blank), all SQL statements of all existing SQL members are explained.

A query number must be specified, so that with each issued EXPLAIN command, the newly created explanation is added to the appropriate query number. The default query number is 1.

To issue the LISTSQL command, enter function code "X" and specify a valid member name and query number; all SQL statements contained in the specified member(s) are explained.

If LISTSQLB is executed online, the following screen informs you about the processing status of the command and if any errors have occurred.

```
16:55:24              ***** NATURAL Tools for SQL *****            1999-09-30
                                  - LISTSQLB -
    Queryno : 1                              Member  Stmtno Message

    Current Object :
    Library         TEST
    Member          RTTB--IN

    Statistics :
    Members read    1
       with SQL     1
    SQL statements  7

       Member   Message




       RTTB--IN OK

Press ENTER to continue
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
```

If executed in batch mode, error messages are written to a dataset referred to by DD name CMPRINT (logical printer 0).

SQLERR Command

The SQLERR command is used to obtain diagnostic information on a DB2 error.

When a DB2 error occurs, Natural issues an appropriate error message. When you enter the SQLERR command, the following information on the most recent DB2 error is displayed:

- the Natural error message number;
- the corresponding reason code (if applicable);
- the variables SQLSTATE and SQLCODE returned by DB2;
- the DB2 error message.

The SQLERR command can be issued either from the Natural NEXT prompt or from within a Natural program (by using the FETCH statement).

Sample SQLERR Diagnostic Information Screen

```
                ***** SQLERR Diagnostic Information *****
    ---------------------- NATURAL SQL Interface Codes -------------------------
    Return Code: 3700    Reason Code: 0      SQLSTATE : 52003   SQL code : -206
    --------------------------------- SQLCA-----------------------------------
    SQLERRP (DB2 Sub routine where error occurred)              :  DSNXOGP
    SQLERRD (DB2 Internal State)
            RDS Return Code                                    :         700
            DBSS Return Code                                   :           0
            Number of Rows Processed                           :           0
            Estimated Cost                                     :        11.2
            Syntax error on PREPARE or EXECUTE IMMEDIATE       :           0
            Buffer Manager ERROR Code                          :           0
    SQLWARN (Warning Flags)
            Data truncated
            Null Values ignored (AVG,SUM,MAX,MIN)              :
            No. of columns greater than no. of host variables  :
            UPDATE/DELETE without WHERE clause                 :
            SQL Statement not valid in DB2                     :
            Adjustment to DATE/TIMESTAMP Variable made         :
    DB2 Error Message :
    DSNT4081 SQLCODE = -206, ERROR: THE OBJECT TABLE OR VIEW OF THE INSERT,
             DELETE, OR UPDATE STATEMENT IS ALSO IDENTIFIED IN A FROM CLAUSE
```


LISTDBRM Command

The LISTDBRM command is used to display either existing DBRMs of Natural programs or Natural programs referencing a given DBRM.

Important:
LISTDBRM has to be issued from the Natural system library SYSDB2, which means you have to LOGON to SYSDB2 first and then enter the command LISTDBRM.

The following menu is displayed:

```
16:53:20              ***** NATURAL Tools for SQL *****            1999-09-30
                                 - List DBRM -




                       Code Function

                        D   Display DBRMs of Programs
                        R   List Programs Referencing DBRM
                        ?   Help
                        .   Exit

                  Code .. _   Library .. EXAMPLE_
                              Member  .. ________
                              DBRM ..... ________



Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                  Canc
```

The following functions are available:

Code	Description
D	Displays programs with DB2 access and their corresponding DBRM. If no DBRM name is shown, the corresponding program uses dynamic SQL.
R	Lists all programs that use a given DBRM. If no DBRM name is specified, all programs that use dynamic SQL are listed.

The following parameters apply:

Parameter	Description
Library	Specifies the name of a Natural library. Library names beginning with "SYS" are not permitted.This parameter must be specified.
Member	Specifies the name of the Natural program (member) to be displayed. This parameter is optional and can be used to limit the output. If a value is specified followed by an asterisk (*), all members in the specified library with names beginning with this value are listed. If the Member field is left blank, or if an asterisk is specified only, all members in the specified library are listed.
DBRM	Specifies a valid DBRM name. If left blank, programs that run dynamically are referenced. This parameter applies to function code "R" only.

Sample List DBRM Result Screen

```
16:53:20               ***** LISTDBRM Command *****               1999-04-26

    Library  Name     Type         DBRM       User ID   Date       Time
    -------- -------- -----------  --------   --------  --------   --------
    EXAMPLE  PROG1    Program      PACK1      SAG       1996-07-17 11:10:43
    EXAMPLE  PROG2    Program      PACK1      SAG       1996-07-17 11:10:48
    EXAMPLE  PROG3    Program      PACK2      SAG       1995-07-17 11:11:04
    EXAMPLE  PROG4    Program                 SAG       1993-07-17 11:11:07
```


Natural Tools for DB2 with Natural Security

The use of the Natural Tools for DB can be restricted by Natural Security:

- You can restrict the access to the the Natural system library SYSDB2.
- Also, you can disallow individual functions by disallowing modules within the library SYSDB2. The following modules can be disallowed:

Module	Function
APMENU-P	Application plan maintenance.
APENV--P	Job Profile menu of application plan maintenance.
CMMENU-P	Catalog maintenance.
PC MENU-P	Procedure maintenance
ISMENU-P	Interactive SQL.
RTMENU-P	Retrieval of system tables.
ESMENU-P	Environment setting.
XPMENU-P	Explain PLAN_TABLE.
SEMENU-P	File server statistics.

In addition, the following parameter settings are recommended for the security profile of the Natural system library SYSDB2:

Parameter	Value
Startup	MENU
Batch Execution	NO

NDB - DDM Generation

This section covers the following topics:

- Natural Data Definition Module - DDM
- SQL Services

Natural Data Definition Module - DDM

To enable Natural to access a DB2 table, a logical Natural Data Definition Module (DDM) of the table must be generated. This is done either with Predict (see the relevant Predict documentation for details) or with the Natural utility SYSDDM.

If you do not have Predict installed, use the SYSDDM function SQL Services to generate Natural DDMs from DB2 tables. This function is invoked from the main menu of SYSDDM and is described on the following pages.

SQL Services

The SQL Services function of the Natural SYSDDM utility (see the relevant documentation) is used to access DB2 tables. You access the catalog of the DB2 server to which you are connected, for example, by using the Environment Setting function as described in Natural Tools for DB2. The name of the DB2 server to which you are connected is then displayed in the top left-hand corner of the screen SQL Services Menu. You can access any DB2 server that is located on either a mainframe or a UNIX platform if the servers have been connected via DRDA (Distributed Relational Database Architecture). For further details on connecting DB2 servers and for information on binding the application package (SYSDDM uses I/O module NDBIOMO) to access data on remote servers, refer to the relevant IBM literature.

The SQL Services function determines whether you are connected to a mainframe DB2 or a UNIX DB2, access the appropriate DB2 catalog and performs the functions listed below.

If you use SYSDDM SQL services in a CICS environment without file server, specify CONVERS=ON in the NDBPARM module (see the relevant section in Installing Natural for DB2); otherwise you might get SQL code -518.

If you select SQL Services on the main menu of the SYSDDM utility, a menu is displayed, which offers you the following functions:

- Select SQL Table from a List
- Generate DDM from an SQL Table
- List Columns of an SQL Table

Select SQL Table from a List

This function is used to select a DB2 table from a list for further processing.

To invoke the function, enter Function Code **S** on the SQL Services Menu.

If you enter the function code only, you obtain a list of all tables defined to the DB2 catalog.

If you do not want a list of all tables but would like only a certain range of tables to be listed, you can, in addition to the function code, specify a start value in the Table Name and/or Creator fields. You can also use asterisk notation (*) for the start value.

When you invoke the function, the Select SQL Table From A List screen is invoked displaying a list of all DB2 tables requested.

On the list, you can mark a DB2 table with either **G** for Generate DDM from an SQL Table or **L** for List Columns of an SQL Table. Then the corresponding function is invoked for the marked table.

Generate DDM from an SQL Table

This function is used to generate a Natural DDM from a DB2 table, based on the definitions in the DB2 catalog.

To invoke the function, enter Function Code **G** on the SQL Services Menu along with the name and creator of the table for which you wish a DDM to be generated.

If you do not know the table name/creator, you can use the function Select SQL Table from a List to choose the table you want.

If you do not want the creator of the table to be part of the DDM name, enter an **N** in the field DDM Name with Creator when you invoke the Generate function (default is **Y**).

Important:
Since the specification of any special characters as part of a field or DDM name does not comply with Natural naming conventions, any special characters allowed within DB2 must be avoided.
DB2 delimited identifiers must be avoided, too.

If you wish to generate a DDM for a table for which a DDM already exists and you want the existing one to be replaced by the newly generated one, enter a **Y** in the Replace field when you invoke the Generate function.

By default, Replace is set to **N** to prevent an existing DDM from being replaced accidentally. If Replace is **N**, you cannot generate another DDM for a table for which a DDM has already been generated.

DBID/FNR Assignment

When the function Generate DDM from an SQL Table is invoked for a table for which a DDM is to be generated for the first time, the DBID/FNR Assignment screen is displayed. If a DDM is to be generated for a table for which a DDM already exists, the existing DBID and FNR are used and the DBID/FNR Assignment screen is suppressed.

On the DBID/FNR Assignment screen, enter one of the database IDs (DBIDs) chosen at Natural installation time, and the file number (FNR) to be assigned to the DB2 table. Natural requires these specifications for identification purposes only.

The range of DBIDs which is reserved for DB2 tables is specified in the NTDB macro of the Natural parameter module (see the Natural Parameter Reference documentation) in combination with the NDBID macro of the parameter module NDBPARM. Any DBID not within this range is not accepted. The FNR can be any valid file number within the database (between 1 and 255).

After a valid DBID and FNR have been assigned, a DDM is automatically generated from the specified table.

Long Field Redefinition

The maximum field length supported by Natural is 1 GB-1 (1073741823 bytes). If a DB2 table contains a column which is longer than 253 bytes or if a DB2 column is defined as a DB2 LOB field, the pop-up window Long Field Generation will be invoked automatically. A DB2 LOB field may be defined as a simple Natural variable with a maximum length of 1GB-1, or as a dynamic Natural variable.

A field which is longer than 253 bytes and which is not a DB2 LOB field may be defined as a simple Natural field with a maximum length of 1GB-1, or as an array. In the DDM, such an array is represented as a multiple-value variable.

If, for example, a DB2 column has a length of 2000 bytes, you can specify an array element length of 200 bytes, and you receive a multiple-value field with 10 occurrences, each occurrence with a length of 200 bytes.

Since redefined long fields are not multiple-value fields in the sense of Natural, the Natural C* notation makes no sense here and is therefore not supported.

When such a redefined long field is defined in a Natural view to be referenced by Natural SQL statements (that is, by host variables which represent multiple-value fields), both when defined and when referenced, the specified range of occurrences (index range) must always start with occurrence 1. If not, a Natural syntax error is returned.

Example:

```
UPDATE table SET varchar = #arr(*)
SELECT ... INTO #arr(1:5)
```

Note:
When such a redefined long field is updated with the Natural DML UPDATE statement (see the relevant section in Statements and System Variables), care must be taken to update each occurrence appropriately.

Length Indicator for Variable Length Fields: VARCHAR, LONG VARCHAR, VARGRAPHIC, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB

For each of the columns listed above, an additional length indicator field (format/length I2 or I4 for LOB fields) is generated in the DDM. The length is always measured in number of characters, not in bytes. To obtain the number of bytes of a VARGRAPHIC, LONG VARGRAPHIC or DBCLOB field, the length must be multiplied by 2.

The name of a length indicator field begins with L@ followed by the name of the corresponding field. The value of the length indicator field can be checked or updated by a Natural program.

If the length indicator field is not part of the Natural view and if the corresponding field is a redefined long field, the length of this field with UPDATE and STORE operations is calculated without trailing blanks.

Null Values

With Natural, it is possible to distinguish between a null value and the actual value zero (0) or blank in a DB2 column.

When a Natural DDM is generated from the DB2 catalog, an additional NULL indicator field is generated for each column which can be NULL; that is, which has neither NOT NULL nor NOT NULL WITH DEFAULT specified.

The name of the NULL indicator field begins with N@ followed by the name of the corresponding field.

When the column is read from the database, the corresponding indicator field contains either zero (0) (if the column contains a value, including the value 0 or blank) or **-1** (if the column contains no value).

Example:

The column NULLCOL CHAR(6) in a DB2 table definition would result in the following view fields:

NULLCOL	**A 6.0**
N@NULLCOL	**I 2.0**

When the field NULLCOL is read from the database, the additional field N@NULLCOL contains:

- 0 (zero) if NULLCOL contains a value (including the value **0** or blank),
- -1 (minus one) if NULLCOL contains no value.

A null value can be stored in a database field by entering **-1** as input for the corresponding NULL indicator field.

Note:
If a column is NULL, an implicit RESET is performed on the corresponding Natural field.

List Columns of an SQL Table

This function lists all columns of a specific DB2 table.

To invoke this function, enter Function Code **L** on the SQL Services Menu along with the name and creator of the table whose columns you wish to be listed.

The List Columns screen for this table is invoked, which lists all columns of the specified table and displays the following information for each column:

Variable	Content
Name	The DB2 name of the column.
Type	The column type.
Length	The length (or precision if Type is DECIMAL) of the column as defined in the DB2 catalog.
Scale	The decimal scale of the column (only applicable if Type is DECIMAL).
Update	Y The column can be updated. N The column cannot be updated.
Nulls	Y The column can contain null values. N The column cannot contain null values.
Not	A column which is of a scale length or type not supported by Natural is marked with an asterisk (*). For such a column, a view field cannot be generated. The maximum scale length supported is 7 bytes. Types supported are: CHAR, VARCHAR, LONG VARCHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DECIMAL, INTEGER, SMALLINT, DATE, TIME, TIMESTAMP, FLOAT, ROWID, BLOB, CLOB and DBCLOB.

The data types DATE, TIME, TIMESTAMP, FLOAT and ROWID are converted into numeric or alphanumeric fields of various lengths: DATE is converted into A10, TIME into A8, TIMESTAMP into A26, FLOAT into F8 and ROWID into A40.

For DB2, Natural provides a DB2 TIMESTAMP column as an alphanumeric field (A26) in the format *YYYY-MM-DD-HH.SS.MMMMMM*.

Since Natural does not yet support computations with such fields, a Natural subprogram called NDBSTMP is provided to enable this kind of functionality.

Dynamic and Static SQL Support

This section covers the following topics:

- General Information
- Internal Handling of Dynamic Statements
- Preparing Natural Programs for Static Execution
- Assembler/Natural Cross-References
- Execution of Natural in Static Mode
- Static SQL with Natural Security
- Mixed Dynamic/Static Mode
- Messages and Codes
- Application Plan Switching in Static SQL

For an explanation of the symbols used in this section to describe the syntax of Natural statements, see Syntax Symbols in the Natural Statements documentation.

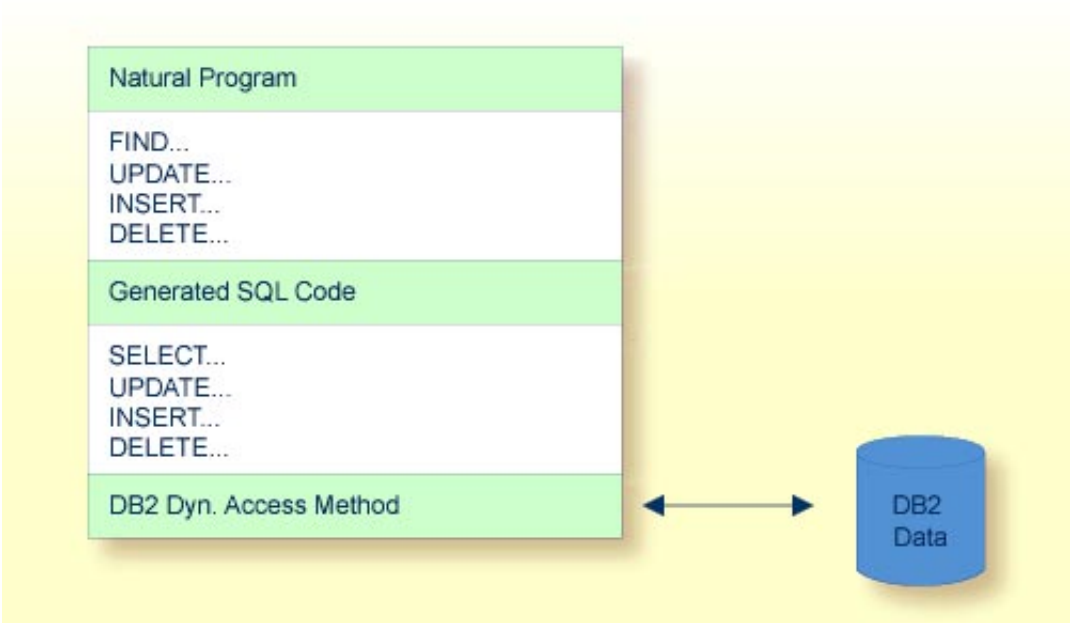
General Information

The SQL support of Natural combines the flexibility of dynamic SQL support with the high performance of static SQL support.

In contrast to static SQL support, the Natural dynamic SQL support does not require any special consideration with regard to the operation of the SQL interface. All SQL statements required to execute an application request are generated automatically and can be executed immediately with the Natural RUN command. Before executing a program, you can look at the generated SQL code, using the LISTSQL command.

Access to DB2 through Natural has the same form whether dynamic or static SQL support is used. Thus, with static SQL support, the same SQL statements in a Natural program can be executed in either dynamic or static mode. An SQL statement can be coded within a Natural program and, for testing purposes, it can be executed using dynamic SQL. If the test is successful, the SQL statement remains unchanged and static SQL for this program can be generated.

Thus, during application development, the programmer works in dynamic mode and all SQL statements are executed dynamically, whereas static SQL is only created for applications that have been transferred to production status.



Internal Handling of Dynamic Statements

Natural automatically provides for the preparation and execution of each SQL statement and handles the opening and closing of cursors used for scanning a table.

This section covers the following topics:

- NDBIOMO
- Statement Table
- Processing of SQL Statements Issued by Natural

NDBIOMO

As each dynamic execution of an SQL statement requires a statically defined DECLARE STATEMENT and DECLARE CURSOR statement, a special I/O module (NDBIOMO) is provided which contains a fixed number of these STATEMENTs and CURSORs. This number is specified during the generation of NDBIOMO (see Step 2 in Steps Common to all Environments, Installing Natural for DB2).

Statement Table

If possible, an SQL statement is only prepared once and can then be executed several times if required. For this purpose, Natural internally maintains a table of all SQL statements that have been prepared and assigns each of these statements to a DECLAREd STATEMENT in NDBIOMO. In addition, this table maintains the cursors used by the SQL statements SELECT, FETCH, UPDATE (Positioned), and DELETE (Positioned).

Each SQL statement is uniquely identified by:

- the name of the Natural program that contains this SQL statement,
- the line number of the SQL statement in this program,
- the name of the Natural library, into which this program was stowed,
- the time stamp when this program was stowed.

Once a statement has been prepared, it can be executed several times with different variable values, using the dynamic SQL statement EXECUTE USING DESCRIPTOR or OPEN CURSOR USING DESCRIPTOR respectively.

When the full capacity of the statement table is reached, the entry for the next prepared statement overwrites the entry for a free statement whose latest execution is the least recent one.

When a new SELECT statement is requested, a free entry in the statement table with the corresponding cursor is assigned to it and all subsequent FETCH, UPDATE, and DELETE statements referring to this SELECT statement will use this cursor. Upon completion of the sequential scanning of the table, the cursor is released and free for another assignment. While the cursor is open, the entry in the statement table is marked as used and cannot be reused by another statement.

If the number of nested FIND (SELECT) statements reaches the number of entries available in the statement table, any further SQL statement is rejected at execution time and a Natural error message is returned.

The size of the statement table depends on the size specified for NDBIOMO. Since the statement table is contained in the DB2 buffer area, the DB2SIZE parameter (see Natural Parameter Modification for DB2, Installing Natural for DB2) may not be sufficient and may need to be increased.

Processing of SQL Statements Issued by Natural

The embedded SQL uses cursor logic to handle SELECT statements. The preparation and execution of a SELECT statement is done as follows:

1. The typical SELECT statement is prepared by a program flow which contains the following embedded SQL statements (note that *X* and *SQLOBJ* are SQL variables, not program labels):

   ```
   DECLARE SQLOBJ STATEMENT

   DECLARE X CURSOR FOR SQLOBJ

   INCLUDE SQLDA (copy SQL control block)

   Then, the following statement is moved into SQLSOURCE:

   SELECT PERSONNEL_ID, NAME, AGE FROM EMPLOYEES WHERE NAME IN (?, ?)    AND AGE BETWEEN ? AND ?
   ```

   (The question marks above are parameter markers which indicate where values are to be inserted at execution time.)

   ```
   PREPARE SQLOBJ FROM SQLSOURCE
   ```

2. Then, the SELECT statement is executed as follows:
   ```
   OPEN X USING DESCRIPTOR SQLDA
   FETCH X USING DESCRIPTOR SQLDA
   ```
   The descriptor SQLDA is used to indicate a variable list of program areas. When the OPEN statement is executed, it contains the address, length, and type of each value which replaces a parameter marker in the WHERE clause of the SELECT statement. When the FETCH statement is executed, it contains the address, length, and type of all program areas which receive fields read from the table.
   When the FETCH statement is executed for the first time, it sets the Natural system variable *NUMBER to a non-zero value if at least one record is found that meets the search criteria. Then, all records satisfying the search criteria are read by repeated execution of the FETCH statement.
   To help improve performance, especially when using distributed databases, the DB2-specific FOR FETCH ONLY clause can be used. FOR FETCH ONLY is generated and executed if rows are to be retrieved only; that is, if no updating is to take place.
3. Once all records have been read, the cursor is released by executing the following statement:
   ```
   CLOSE X
   ```


Preparing Natural Programs for Static Execution

This section covers the following topics:

- Basic Principles
- Generation Procedure - CMD CREATE Command
- Precompilation of the Generated Assembler Program
- Modification Procedure - CMD MODIFY Command
- BIND of the Precompiled DBRM

Basic Principles

Static SQL is generated in Natural batch mode for one or more Natural applications which can consist of one or more Natural object programs. The number of programs that can be modified for static execution in one run of the generation procedure is limited to 999.

During the generation procedure, the database access statements contained in the specified Natural objects are extracted, written to work files, and transformed into a temporary Assembler program. If no Natural program is found that contains SQL access or if any error occurs during static SQL generation, batch Natural terminates and condition code 40 is returned, which means that all further JCL steps must no longer be executed.

The Natural modules NDBCHNK and NDBSTAT must reside in a steplib of the generation step. Both are loaded dynamically during the execution of the generation step.

The temporary Assembler program is written to a temporary file (the Natural work file CMWKF06) and precompiled. The size of the workfile is proportional to the maximum number of programs, the number of SQL statements and the number of variables used in the SQL statements. During the precompilation step, a database request module (DBRM) is created, and after the precompilation step, the precompiler output is extracted from the Assembler program and written to the corresponding Natural objects, which means that the Natural objects are modified (prepared) for static execution. The temporary Assembler program is no longer used and deleted.

The Natural subprogram NDBDBRM can be used to check whether a Natural program contains SQL access and whether it has been modified for static execution.

A static DBRM is created by using either the sample job provided on the installation tape or an appropriate job created with the Create DBRM function.

Generation Procedure: CMD CREATE Command

To generate static SQL for Natural programs, LOGON to the Natural system library SYSDB2.

Since a new SYSDB2 library has been created when installing Natural for DB2, ensure that it contains all Predict interface programs necessary to run the static SQL generation. These programs are loaded into SYSDB2 at Predict installation time (see the relevant Predict documentation).

Then specify the CMD CREATE command and the Natural input necessary for the static SQL generation process; the CMD CREATE command has the following syntax:

CMD CREATE DBRM *static-name* **USING** *using-clause*

{*application-name,object-name,excluded-object***}**

:

:

The generation procedure reads but does not modify the specified Natural objects. If one of the specified programs was not found or had no SQL access, return code 4 is returned at the end of the generation step.

Static Name

If the PREDICT DOCUMENTATION option is to be used, a corresponding Predict static SQL entry must be available and the *static-name* must correspond to the name of this entry. In addition, the *static-name* must correspond to the name of the DBRM to be created during precompilation. The *static-name* can be up to 8 characters long and must conform to Assembler naming conventions.

USING Clause

The *using-clause* specifies the Natural objects to be contained in the DBRM. These objects can either be specified explicitly as INPUT DATA in the JCL or obtained as PREDICT DOCUMENTATION from Predict.

{ INPUT DATA | PREDICT DOCUMENTATION } [WITH XREF { YES | NO | FORCE }] [FS { OFF | ON }] [LIB *lib-name***]**

If the parameters to be specified do not fit in one line, specify the command identifier (CMD) and the various parameters in separate lines and use both the input delimiter (as specified with the ID parameter; default is a comma (,)) and the continuation indicator (as specified with the CF parameter; default is a percent (%) character as shown in the following example:

Example:

```
CMD
CREATE,DBRM,static,USING,PREDICT,DOCUMENTATION,WITH,XREF,NO,%
LIB,library
```

Alternatively, you can also use abbreviations as shown in the following example:

Example:


```
CMD CRE DBRM static US IN DA W XR Y FS OFF LIB library
```

The sequence of the parameters USING, WITH, FS, and LIB is optional.

INPUT DATA

As input data, the applications and names of the Natural objects to be included in the DBRM must be specified in the subsequent lines of the job stream (*application-name*,*object-name*). A subset of these objects can also be excluded again (*excluded-objects*). Objects in libraries whose names begin with SYS can be used for static generation, too.

The applications and names of Natural objects must be separated by the input delimiter (as specified with the ID parameter; default is a comma (,)). If you wish to specify all objects whose names begin with a specific string of characters, use an *object-name* or *excluded-objects* name that ends with asterisk notation (*). To specify all objects in an application, use asterisk notation only.

Example:

```
LIB1,ABC*
LIB2,A*,AB*
LIB2,*
:
.
```

The specification of applications/objects must be terminated by a line that contains a period (.) only.

PREDICT DOCUMENTATION

Since Predict supports static SQL for DB2, you can also have Predict supply the input data for creating static SQL by using already existing PREDICT DOCUMENTATION.

WITH XREF Option

Since Predict Active References supports static SQL for DB2, the generated static DBRM can be documented in Predict and the documentation can be used and updated with Natural.

WITH XREF is the option which enables you to store cross-reference data for a static SQL entry in Predict each time a static DBRM is created (YES). You can instead specify that no cross-reference data are stored (NO) or that a check is made to determine whether a Predict static SQL entry for this static DBRM already exists (FORCE). If so, cross-reference data are stored; if not, the creation of the static DBRM is not allowed. For more detailed information on Predict Active References, refer to the relevant Predict documentation.

When WITH XREF (YES/FORCE) is specified, XREF data are written for both the Predict static SQL entry (if defined in Predict) and each generated static Natural program. However, static generation with WITH XREF (YES/FORCE) is possible only if the corresponding Natural programs have been cataloged with XREF ON.

WITH XREF FORCE only applies to the USING INPUT DATA option.

Note:
If you do not use Predict, the XREF option must be omitted or set to NO and the module NATXRF2 need not be linked to the Natural nucleus.

FS Option

If the FS (file server) option is set to ON, a second SELECT is generated for the Natural file server. ON is the default setting.

If the FS option is set to OFF, no second SELECT is generated, which results in less SQL statements being generated in your static DBRM and thus in a smaller DBRM.

LIB Option

With the LIB (library) option, a Predict library other than the default library (*SYSSTA*) can be specified to contain the Predict static SQL entry and XREF data. The name of the library can be up to eight characters long.

Precompilation of the Generated Assembler Program

In this step, the precompiler is invoked to precompile the generated temporary Assembler program. The precompiler output consists of the DBRM and a precompiled temporary Assembler program which contains all the database access statements transformed from SQL into Assembler statements.

Later, the DBRM serves as input for the BIND step and the Assembler program as input for the modification step.

Modification Procedure: CMD MODIFY Command

The modification procedure modifies the Natural objects involved by writing precompiler information into the object and by marking the object header with the *static-name* as specified with the CMD CREATE command.

In addition, any existing copies of these objects in the Natural global buffer pool (if available) are deleted and XREF data are written to Predict (if specified during the generation procedure).

To perform the modification procedure, LOGON to the Natural system library SYSDB2 and specify the CMD MODIFY command which has the following syntax:

```
CMD MODIFY  [ XREF ]
```

The input for the modify step is the precompiler output which must reside on a dataset defined as the Natural work file CMWKF01.

The output consists of precompiler information which is written to the corresponding Natural objects. In addition, a message is returned telling you whether it was the first time an object was modified for static execution (modified) or whether it had been modified before (re-modified).

If the XREF option is specified, the Natural work file CMWKF02 must be defined to contain the resulting list of cross-reference information concerning the statically generated SQL statements (see also Assembler/Natural Cross-References).

BIND of the Precompiled DBRM

It is recommended to run the BIND job **after** the MODIFY job.

This step performs the BIND against DB2. One or more DBRMs (as created by the precompiler) are processed to create a DB2 application plan. In addition to the static DBRMs created above, this application plan contains the dynamic DBRM NDBIOMO provided by Natural itself.

A DBRM can be bound into any number of application plans where it might be required. A plan is physically independent of the environment where the program is to be run. However, you can group your DBRMs logically into plans which are to be used for either batch or online processing, where the same DBRM can be part of both a batch plan and an online plan.

Unless you are using plan switching, only one plan can be executed per Natural session. Thus, you must ensure that the plan name specified in the BIND step is the same as the one used to execute Natural.

Assembler/Natural Cross-References

If you specify the XREF option of the MODIFY command, an output listing is created on the work file CMWKF02, which contains the DBRM name and the Assembler statement number of each statically generated SQL statement together with the corresponding Natural source code line number, program name, library name, database ID and file number.

```
-------------------------------------------------------------------------....
  DBRMNAME STMTNO      LINE NATPROG  NATLIB   DB  FNR COMMENT             ....
-------------------------------------------------------------------------....
  TESTDBRM 000627      0390 TESTPROG SAG      010 042 INSERT              ....
           000641      0430                          INSERT              ....
           000652      0510                          SELECT              ....
           000674      0570                          SELECT              ....
           000698      0570                          SELECT      2ND     ....
           000728      0650                          UPD/DEL             ....
           000738      0650                          UPD/DEL     2ND     ....
           000751      0700                          SELECT              ....
           000775      0700                          SELECT      2ND     ....
```

Column	Explanation
DBRMNAME	Name of the DBRM which contains the static SQL statement.
STMTNO	Assembler statement number of the static SQL statement.
LINE	Corresponding Natural source code line number.
NATPROG	Name of the Natural program that contains the static SQL statement.
NATLIB	Name of the Natural library that contains the Natural program.
DB / FNR	Natural database ID and file number.
COMMENT	Type of SQL statement, where 2ND indicates that the corresponding statement is used for a reselection; see also the Concept of the File Server.

Execution of Natural in Static Mode

To be able to execute Natural in static mode, all users of Natural must have the DB2 EXECUTE PLAN/PACKAGE privilege for the plan created in the BIND step.

To execute static SQL start Natural and execute the corresponding Natural program. Internally, the Natural runtime interface evaluates the precompiler data written to the Natural object and then performs the static accesses.

To the user there is no difference between dynamic and static execution.

Static SQL with Natural Security

Static generation can be disallowed with Natural Security by:

- restricting access to the Natural system library SYSDB2,
- disallowing the module CMD,
- restricting access to the libraries that contain the relevant Natural objects,
- disallowing one of the commands CATALOG, or STOW for a library that contains relevant Natural objects.

If a library is defined in Natural Security and the DBID and FNR of this library are different from the default specifications, the static generation procedure automatically switches to the DBID and FNR specifications defined in Natural Security.

Mixed Dynamic/Static Mode

It is possible to operate Natural in a mixed static and dynamic mode where for some programs static SQL is generated and for some not.

The mode in which a program is run is determined by the Natural object program itself. If a static DBRM is referenced in the executing program, all statements in this program are executed in static mode.

Note:
Natural programs which return a runtime error do not automatically execute in dynamic mode. Instead, either the error must be corrected or, as a temporary solution, the Natural program must be recataloged to be able to execute in dynamic mode.

Within the same Natural session, static and dynamic programs can be mixed without any further specifications. The decision which mode to use is made by each individual Natural program.

Messages and Codes

This section lists the error messages that may be issued during static generation.

STAT9001 Object buffer allocation failed. RC = *return code*

Program NDBCHNK has been invoked to allocate space for Natural object load, but the allocation has failed; retry or increase the free storage pool.

STAT9002 Write on object area failed. RC = *return code*

Program NDBCHNK has been invoked to write a Natural object row into the appropriate buffer, but the write has failed; this is probably a NDBCHNK program error.

STAT9003 Statement entry retrieve error. RC = *return code*

Program NDBSTAT has been invoked to retrieve next DB2 statement information from the Natural object loaded in main storage, but the retrieval has failed (RC was neither 0 (OK) nor 4 (EOP)); the probable cause is a Natural object inconsistency.

STAT9004 Unsupported Adabas command: *command*

Program NDBSTAT has been invoked to retrieve next DB2 statement information from the Natural object loaded in main storage, but the Adabas command code returned was invalid; the probable cause is a Natural object inconsistency.

STAT9005 Freemain failed. RC = *return code*

Program NDBCHNK has been invoked to free the area allocated for Natural object load, but the release has failed; this is probably a program error.

STAT9006 Call for timestamp of program failed. RC = *return code*

Program NDBSTAT has been invoked to know the time stamp associated to the loaded Natural object, but the call has failed; this is probably a program error.

STAT9007 A-List item retrieve failed. RC = *return code*

Program NDBSTAT has been invoked to retrieve the next compilation A-list element, but the retrieval has failed (RC was neither 0 (OK) nor 20 (EOL)); the probable cause is a Natural object inconsistency.

STAT9009 Invalid database field format: *format*

Program NDBSTAT has been invoked to retrieve the next compilation A-list element, but the DB2 format code returned is invalid; the probable cause is a Natural object inconsistency.

STAT9014 Warning, may indicate a problem: second select table reset.

The table for a second selection logs the statement number of all second SELECT statements. The table is reset if there are more than 100 entries, which means with many nested program loops. If the table is reset, no second UPDATE or DELETE statements are generated.

STAT9016 Versions of NDBSTAT and SQLGEN Natural programs do not match.

The versions of the Natural programs used for the static generation (library SYSSQL) must be the same as one of the dynamically loaded Assembler program NDBSTAT.

STAT9017 *address* **of program** *program* **in library** *library* **not found.**

A Natural object address was not found and the object cannot be modified. Either the object was not found or the address was wrong.

STAT9019 * Warning: Natural terminates abnormally, run may continue. *****

Warning: Natural terminates abnormally with RC=4. A Natural member was explicitly entered which does not exist or does not have SQL access. The static generation can continue.

STAT9020 Start run of SQLGEN for DBRM *dbrm***.**

STAT9021 Start merging temporary datasets.

STAT9022 Precompile input *input* **written to temporary dataset.**

The temporary assembler program for the precompiler input was written to a temporary dataset (Natural work file 6).

STAT9023 * END OF DATA *****

STAT9024 No program with SQL access found.

None of the programs processed by the CMD command accessed an SQL system.

STAT9025 Program *program* **in library** *library* **not found.**

STAT9026 DB access module names *module* **and** *module* **do not match.**

The module name specified with the CMD CREATE command must be the same as the name of the DBRM specified in the DBRMLIB job card of the precompilation step.

STAT9027 Error *error* **purging** *program***,** *library* **in buffer pool. Run continues.**

STAT9028 Number of programs to be generated exceeds 999.

The number of programs to be generated statically into one DBRM exceeds the maximum number of 999.

STAT9029 Limit of *limit* **NULL indicators per SQL statement exceeded.**

The maximum number of 1500 NULL indicators per SQL statement has been exceeded.

STAT9030 Number of variables to be generated exceeds 9999.

The number of variables to be generated statically for one program exceeds the maximum number of 9999.

STAT9031 XREF option NO and Predict DDA default "YES" do not match.

The Predict DDA default setting for static SQL XREF is set to "YES" but the XREF option in the CMD command is set to "NO".

STAT9032 XREF option "FORCE" but no Predict documentation found.

With the XREF option FORCE, the static generation continues and writes XREF data only if Predict documentation exists for a given DBRM. If there is no Predict documentation available, static generation is not performed.

STAT9033 No XREF data exist for member *member***.**

Either the Natural program which is to be statically generated cannot be cataloged with XREF=ON or the XREF data are not on the used Predict file.

STAT9034 XREF option "YES" or "NO" but Predict DDA default "FORCE".

The Predict DDA default setting for static SQL XREF is set to "FORCE", but the XREF option in the CMD command is set to "NO" or "YES.

STAT9036 Given DBRM library not defined as 3GL Predict application.

The library for the DBRM entered with the LIB option is not defined as 3GL application in Predict. Check the library name in Predict which contains the DBRM.

STAT9039 Library name must not be blank.

STAT9040 CAT or STOW not allowed for library *library***.**

The commands CAT or STOW are not allowed in your security environment. However, the CAT or STOW privilege is needed for static generation.

STAT9041 Natural Security restriction. Message code: *message code*

STAT9050 No Predict documentation for specified DBRM found.

No documentation was found in Predict for the DBRM specified with the CMD command. Either the DBRM is not documented in the used Predict file or a wrong DBRM name has been specified.

STAT9062 No Predict installed or SM level less than SM4.

STAT9063 XREF interface not linked. XREF option reset, run continues.

STAT9064 XREF option not set. Predict DDA default *default* **taken.**

The Predict DDA default setting for static SQL XREF is read, because no XREF option is specified in the CMD command and the XREF interface and Predict are installed.

STAT9065 DBRM name must start with an uppercase character.

STAT9066 No Predict installed or SM level less than SM4, run continues.

STAT9072 DBRM name must not be blank.

STAT9073 Invalid syntax for *parameter/option* **specified.**

STAT9092 Error occurred. XREF data for DBRM will be deleted.

STAT9093 Error *error* **occurred in program** *program* **on line** *line***.**

STAT9094 Return code *return code* **on call of** *program***.**

STAT9095 Error in *parameter* **parameter on call of** *program***.**

Application Plan Switching in Static SQL

When using application plan switching, you can switch to a different application plan within the same Natural session.

If a second application plan is to be used, this can be specified by executing the Natural program NATPLAN. NATPLAN is contained in the Natural system library SYSDB2 and can be invoked either from within a Natural program or dynamically by entering the command NATPLAN in the NEXT line. The only input value required for NATPLAN is an eight-character plan name. If no plan name is specified, you are prompted by the system to do so.

Before executing NATPLAN, ensure that any open DB2 recovery units are closed.

Since the NATPLAN program is also provided in source form, user-written plan switching programs can be created using similar logic.

The actual switch from one plan to another differs in the various environments supported. The feature is available under Com-plete, CICS, and IMS/TM MPP. When using the CAF interface, it is also available in TSO and batch environments.

In some of these environments, a transaction ID or code must be specified instead of a plan name.

This section covers the following topics:

- Plan Switching under CICS
- Plan Switching under Com-plete
- Plan Switching under IMS/TM
- Plan Switching under TSO and in Batch Mode

Plan Switching under CICS

Under CICS, you have the option of using either plan switching by transaction ID (default) or dynamic plan selection exit routines. Thus, by setting the field #SWITCH-BY- TRANSACTION-ID in the NATPLAN program to either TRUE or FALSE, either the subroutine CMTRNSET or the module NDBPLNA is invoked.

See more information on activating plan switching under CICS in the section Steps Specific to CICS (Installing Natural for DB2).

Plan Switching by Transaction ID

If #SWITCH-BY-TRANSACTION-ID is set to TRUE, the subroutine CMTRNSET is invoked, which changes the current pseudo-conversational transaction ID to the one you want to switch to.

CMTRNSET is documented in the Natural system library SYSEXTP, which is invoked with the SYSEXTP system command. After calling CMTRNSET, you have to perform a terminal I/O to ensure that a new CICS transaction is used.

Using plan switching by transaction ID enables you to use existing CICS entry threads for your Natural transactions, where each transaction ID can have its own entry thread assigned.

Plan Switching by CICS/DB2 Exit Routine

If #SWITCH-BY-TRANSACTION-ID is set to FALSE, the NDBPLNA module is invoked to supply a specified CICS/DB2 exit routine with the plan name to be used. Thus, for a Natural application to perform plan switching by a CICS/DB2 exit routine, the NATPLAN program must be invoked before the first DB2 access. For additional information on CICS/DB2 exit routines, refer to the relevant IBM literature.

NDBPLNA writes a CICS temporary storage record containing the plan name that was supplied to NATPLAN. The name of the temporary storage queue is PLAN*xxxx*, where *xxxx* is the CICS terminal identifier.

When running in a CICSplex environment, the CICS temporary storage queue PLAN*xxxx* containing the plan name must be defined with TYPE=SHARED or TYPE=REMOTE in a CICS TST.

For each new DB2 unit of recovery, the appropriate plan selection exit routine is automatically invoked. This exit routine reads the temporary storage record and uses the contained plan name for plan selection.

When no temporary storage record exists for the Natural session, a default plan name, contained in the plan exit, can be used. If no plan name is specified by the exit, the name of the plan used is the same as the name of the static program (DBRM) issuing the SQL call. If no such plan name exists, an SQL error results.

Plan Switching under Com-plete

In Com-plete environments, plan switching is accomplished using the Call Attachment Facility (CAF) to release the thread in use and attach another one that has a different plan name.

Once the DB2 connection has been established, the same plan name continues to be used until the plan is explicitly changed using the call attachment language interface (DSNALI). For additional information on the CAF interface, refer to the relevant IBM literature.

Under Com-plete, the NATPLAN program first issues an END TRANSACTION statement and then invokes an Assembler module named NDBPLAN by using DB2SERV.

NDBPLAN performs the actual switching. It issues a CLOSE request to DSNALI to terminate the DB2 connection (if one exists). It then issues an OPEN request to re-establish the DB2 connection and to allocate the resources needed to execute the specified plan.

If NATPLAN has not been executed before the first SQL call, the default plan used is the one defined in the Com-plete startup parameters. Once a plan has been changed using NDBPLAN, it remains scheduled until another plan is scheduled by NDBPLAN or until the end of the Natural session.

Plan Switching under IMS/TM

In an IMS MPP environment, the switch is accomplished using direct or deferred message switching. As a different application plan is associated with each IMS application program, message switching from one transaction code to another automatically switches the application plan being used.

Since Natural applications can perform direct or deferred message switches by calling the appropriate supplied routines, use of the NATPLAN program for plan switching is optional.

NATPLAN calls the Assembler routine CMDEFSW, which sets the new transaction code to be used with the next following terminal I/O.

Plan Switching under TSO and in Batch Mode

In the TSO and batch environments, plan switching is accomplished using the Call Attachment Facility (CAF) to release the thread in use and attach another one that has a different plan name. Using CAF, plan selection is either implicit or explicit.

If no DB2 connection has been made before the first SQL call, a plan name is implicitly selected by DB2. If so, the plan name used is the same as the name of the program (DBRM) issuing the SQL call.

Once the DB2 connection has been established, the same plan name continues to be used until the plan is explicitly changed using the call attachment language interface (DSNALI). For additional information on the CAF interface, refer to the relevant IBM literature.

Under TSO and in batch mode, the NATPLAN program first issues an END TRANSACTION statement and then invokes an Assembler module named NDBPLAN by using DB2SERV.

Note:
Modify the NATPLAN program by setting the #SSM field to the current DB2 subsystem name; the default name is DB2.

NDBPLAN performs the actual switching. It issues a CLOSE request to DSNALI to terminate a possible DB2 connection. It then issues an OPEN request to re-establish the DB2 connection and to allocate the resources needed to execute the specified plan.

If NATPLAN has not been executed before the first SQL call, plan selection is done implicitly by DSNALI. If so, the plan name used is the same as the name of the program issuing the SQL call. The subsystem ID used is the one specified during the DB2 installation. If no such plan name or subsystem ID exists, a Natural error message is returned.

If a static DBRM is issuing the SQL call, a plan name must exist with the same name as the one of the static DBRM.

If dynamic SQL is being used, a plan name of NDBIOMO must exist which must contain a DBRM called NDBIOMO, too. NDBIOMO is the default plan name.

Note:
To avoid any confusion concerning the chosen plan name and/or subsystem ID, always call NATPLAN before the first SQL call.

NDB - Statements and System Variables

This section contains special considerations concerning Natural DML statements, Natural SQL statements, and Natural system variables when used with DB2.

It mainly consists of information also contained in the Natural documentation set where each Natural statement and variable is described in detail.

For an explanation of the symbols used in this section to describe the syntax of Natural statements, see Syntax Symbols in the Natural Statements documentation.

This section covers the following topics:

- Natural DML Statements
- Natural SQL Statements
- Natural System Variables
- Error Handling

NDB - Natural DML Statements

This section summarizes particular points you have to consider when using Natural DML statements with DB2. Any Natural statement not mentioned in this section can be used with DB2 without restriction.

- BACKOUT TRANSACTION
- DELETE
- END TRANSACTION
- FIND
- GET
- HISTOGRAM
- READ
- STORE
- UPDATE

BACKOUT TRANSACTION

This statement undoes all database modifications made since the beginning of the last logical transaction. Logical transactions can start either after the beginning of a session or after the last SYNCPOINT, END TRANSACTION, or BACKOUT TRANSACTION statement.

How the statement is translated and which command is actually issued depends on the TP-monitor environment:

- If this command is executed within a Natural stored procedure or Natural user-defined function (UDF), Natural for DB2 executes the underlying rollback operation. This sets the caller into a must-rollback state. If this command is executed within a Natural stored procedure or UDF for Natural error processing (implicit ROLLBACK), Natural for DB2 does not execute the underlying rollback operation, thus allowing the caller to receive the original Natural error.
- Under CICS, the BACKOUT TRANSACTION statement is translated into an EXEC CICS ROLLBACK command. However, in pseudo-conversational mode, only changes made to the database since the last terminal I/O are undone. This is due to CICS-specific transaction processing.
  Note:
  Be aware that with terminal input in database loops, Natural switches to conversational mode if no file server is used.

- In batch mode and under TSO, the BACKOUT TRANSACTION statement is translated into an SQL ROLLBACK command.
  Note:
  If running in a DSNMTV01 environment the BACKOUT TRANSACTION statement is ignored if the used PSB has been generated without the CMPAT=YES option.

- Under IMS/TM, the BACKOUT TRANSACTION statement is translated into an IMS Rollback (ROLB) command. However, only changes made to the database since the last terminal I/O are undone. This is due to IMS/TM-specific transaction processing.

As all cursors are closed when a logical unit of work ends, a BACKOUT TRANSACTION statement must not be placed within a database loop; instead, it has to be placed outside such a loop or after the outermost loop of nested loops.

If an external program written in another standard programming language is called from a Natural program, this external program must not contain its own ROLLBACK command if the Natural program issues database calls, too. The calling Natural program must issue the BACKOUT TRANSACTION statement for the external program.

If a program tries to backout updates which have already been committed, for example by a terminal I/O, a corresponding Natural error message (NAT3711) is returned.

DELETE

The DELETE statement is used to delete a row from a DB2 table which has been read with a preceding FIND, READ, or SELECT statement. It corresponds to the SQL statement DELETE WHERE CURRENT OF *cursor-name*, which means that only the row which was read last can be deleted.

Example:

```
FIND EMPLOYEES WITH NAME = 'SMITH'
     AND FIRST_NAME = 'ROGER'
DELETE
```

Natural would translate the above Natural statements into SQL and assign a cursor name (for example, CURSOR1) as follows:

```
DECLARE CURSOR1 CURSOR FOR
SELECT FROM EMPLOYEES
 WHERE NAME = 'SMITH' AND FIRST_NAME = 'ROGER'
DELETE FROM EMPLOYEES
 WHERE CURRENT OF CURSOR1
```

Both the SELECT and the DELETE statement refer to the same cursor.

Natural translates a DML DELETE statement into an SQL DELETE statement in the same way it translates a FIND statement into an SQL SELECT statement.

A row read with a FIND SORTED BY cannot be deleted due to DB2 restrictions explained with the FIND statement. A row read with a READ LOGICAL cannot be deleted either.

DELETE when using the File Server

If a row rolled out to the file server is to be deleted, Natural rereads automatically the original row from the database to compare it with its image stored in the file server. If the original row has not been modified in the meantime, the DELETE operation is performed. With the next terminal I/O, the transaction is terminated, and the row is deleted from the actual database.

If the DELETE operates on a scrollable cursor, the row on the file server is marked as DELETE hole and is deleted from the base table.

However, if any modification is detected, the row will not be deleted and Natural issues the NAT3703 error message for non-scrollable cursors.

If the DELETE operates on a scrollable cursor, NDB simulates SQLCODE -224 (THE RESULT TABLE DOES NOT AGREE WITH THE BASE TABLE USING) for DB2 compliance.

If the DELETE operates on a scrollable cursor and the row has become a hole, NDB simulates SQLCODE Œ222 (AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST A HOLE).

Since a DELETE statement requires that Natural rereads a single row, a unique index must be available for the respective table. All columns which comprise the unique index must be part of the corresponding Natural view.

END TRANSACTION

This statement indicates the end of a logical transaction and releases all DB2 data locked during the transaction. All data modifications are committed and made permanent.

How the statement is translated and which command is actually issued depends on the TP-monitor environment:

- If this command is executed from a Natural stored procedure or UDF, Natural for DB2 does not execute the underlying commit operation. This allows the stored procedure or UDF to commit updates against non DB2 databases.

- Under CICS, the END TRANSACTION statement is translated into an EXEC CICS SYNCPOINT command. If the file server is used, an implicit end-of-transaction is issued after each terminal I/O. This is due to CICS-specific transaction processing in pseudo-conversational mode.
- In batch mode and under TSO, the END TRANSACTION statement is translated into an SQL COMMIT WORK command.
  Note:
  If running in a DSNMTV01 environment the END TRANSACTION statement is ignored if the used PSB has been generated without the CMPAT=YES option.

- Under IMS/TM, the END TRANSACTION statement is not translated into an IMS CHKP call, but is ignored. Due to IMS/TM-specific transaction processing, an implicit end-of-transaction is issued after each terminal I/O.

Except when used in combination with the SQL WITH HOLD clause (see SELECT in Natural SQL Statements), an END TRANSACTION statement must not be placed within a database loop, since all cursors are closed when a logical unit of work ends. Instead, it has to be placed outside such a loop or after the outermost loop of nested loops.

If an external program written in another standard programming language is called from a Natural program, this external program must not contain its own COMMIT command if the Natural program issues database calls, too. The calling Natural program must issue the END TRANSACTION statement for the external program.

Note:
With DB2, the END TRANSACTION statement cannot be used to store transaction data.

FIND

The FIND statement corresponds to the SQL SELECT statement.

Example:

Natural statements:

```
FIND EMPLOYEES WITH NAME = 'BLACKMORE'
   AND AGE EQ 20 THRU 40
OBTAIN PERSONNEL_ID NAME AGE
```

Equivalent SQL statement:

```
SELECT PERSONNEL_ID, NAME, AGE
  FROM EMPLOYEES
    WHERE NAME = 'BLACKMORE'
      AND AGE BETWEEN 20 AND 40
```


Natural internally translates a FIND statement into an SQL SELECT statement. The SELECT statement is executed by an OPEN CURSOR statement followed by a FETCH command. The FETCH command is executed repeatedly until either all records have been read or the program flow exits the FIND processing loop. A CLOSE CURSOR command ends the SELECT processing.

The WITH clause of a FIND statement is converted to the WHERE clause of the SELECT statement. The basic search criterion for a DB2 table can be specified in the same way as for an Adabas file. This implies that only database fields which are defined as descriptors can be used to construct basic search criteria and that descriptors cannot be compared with other fields of the Natural view (that is, database fields) but only with program variables or constants.

Note:
As each database field (column) of a DB2 table can be used for searching, any database field can be defined as a descriptor in a Natural DDM.

The WHERE clause of the FIND statement is evaluated by Natural **after** the rows have been selected via the WITH clause. Within the WHERE clause, non-descriptors can be used and database fields can be compared with other database fields.

Note:
DB2 does not have sub-, super-, or phonetic descriptors.

A FIND NUMBER statement is translated into a SELECT statement containing a COUNT(*) clause. The number of rows found is returned in the Natural system variable *NUMBER as described in Natural System Variables.

The FIND UNIQUE statement can be used to ensure that only one record is selected for processing. If the FIND UNIQUE statement is referenced by an UPDATE statement, a non-cursor (Searched) UPDATE operation is generated instead of a cursor-oriented (Positioned) UPDATE operation. Therefore, it can be used if you want to update a DB2 primary key. It is, however, recommended to use Natural SQL Searched UPDATE statement to update a primary key.

In static mode, the FIND NUMBER and FIND UNIQUE statements are translated into a SELECT SINGLE statement as described in the section Natural SQL Statements.

The FIND FIRST statement cannot be used. The PASSWORD, CIPHER, COUPLED and RETAIN clauses cannot be used either.

The SORTED BY clause of a FIND statement is translated into the SQL SELECT ... ORDER BY clause, which follows the search criterion. Because this produces a read-only result table, a row read with a FIND statement that contains a SORTED BY clause cannot be updated or deleted.

A limit on the depth of nested database loops can be specified at installation time. If this limit is exceeded, a Natural error message is returned.

Note for DB2 for OS/390 Version 7:
If a processing limit is specified as a constant integer number, for example, FIND (5), the limitation value will be translated into a FETCH FIRST integer ROWS ONLY clause in the generated SQL string.

FIND when using the File Server

As far as the file server is concerned, there are no programming restrictions with selection statements. It is, however, recommended to make yourself familiar with its functionality considering performance and file server space requirements.

GET

This statement is ISN-based and therefore cannot be used with DB2 tables.

HISTOGRAM

The HISTOGRAM statement returns the number of rows in a table which have the same value in a specific column. The number of rows is returned in the Natural system variable *NUMBER as described in Natural System Variables.

Example:

Natural statements:

```
HISTOGRAM EMPLOYEES FOR AGE
OBTAIN AGE
```

Equivalent SQL statement:

```
SELECT COUNT(*), AGE FROM EMPLOYEES
  WHERE AGE > -999
  GROUP BY AGE
  ORDER BY AGE
```

Natural translates the HISTOGRAM statement into an SQL SELECT statement, which means that the control flow is similar to the flow explained for the FIND statement.

READ

The READ statement can also be used to access DB2 tables. Natural translates a READ statement into an SQL SELECT statement.

READ PHYSICAL and READ LOGICAL can be used; READ BY ISN, however, cannot be used, as there is no DB2 equivalent to Adabas ISNs. The PASSWORD and CIPHER clauses cannot be used either.

Since a READ LOGICAL statement is translated into a SELECT ... ORDER BY statement - which produces a read-only table -, a row read with a READ LOGICAL statement cannot be updated or deleted (see Example 1). The start value can only be a constant or program variable; any other field of the Natural view (that is, any database field) cannot be used.

A READ PHYSICAL statement is translated into a SELECT statement without an ORDER BY clause and can therefore be updated or deleted (see Example 2).

Example 1:

Natural statements:

```
READ PERSONNEL BY NAME
OBTAIN NAME FIRSTNAME DATEOFBIRTH
```

Equivalent SQL statement:

```
SELECT NAME, FIRSTNAME, DATEOFBIRTH FROM PERSONNEL
  WHERE NAME >= ' '
    ORDER BY NAME
```

Example 2:

Natural statements:

```
READ PERSONNEL PHYSICAL
OBTAIN NAME
```

Equivalent SQL statement:

```
SELECT NAME FROM PERSONNEL
```

If the READ statement contains a WHERE clause, this clause is evaluated by the Natural processor **after** the rows have been selected according to the descriptor value(s) specified in the search criterion.

NDB Features under DB2 for OS390 Version 7

Processing Limit

If a processing limit is specified as a constant integer number, for example, READ (5), in the SQL string generated, the value that defines the limitation will be translated into the clause

FETCH FIRST *integer* **ROWS ONLY**

Cursors for DB2 Clauses

NDB uses insensitive scrollable cursors to process the following READ statement:

READ .. [IN] [LOGICAL] VARIABLE/DYNAMIC *operand5* **[SEQUENCE]**

NDB uses insensitive scrollable cursors to process the READ statement below. If relating to a Positioned UPDATE or DELETE, NDB uses insensitive static cursors.

READ .. [IN] [PHYSICAL] DESCENDING/VARIABLE/DYNAMIC *operand5* **[SEQUENCE]**

operand5:
Value 'A' will be translated into a FETCH FIRST/NEXT DB2 access, and Value 'D' into a FETCH LAST/PRIOR DB2 access.

READ when using the File Server

As far as the file server is concerned there are no programming restrictions with selection statements. It is, however, recommended to make yourself familiar with its functionality considering performance and file server space requirements.

STORE

The STORE statement is used to add a row to a DB2 table. The STORE statement corresponds to the SQL statement INSERT.

Example:

Natural statement:

```
STORE RECORD IN EMPLOYEES
  WITH PERSONNEL_ID = ’2112’
       NAME         = ’LIFESON’
       FIRST_NAME   = ’ALEX’
```

Equivalent SQL statement:

```
INSERT INTO EMPLOYEES (PERSONNEL_ID, NAME, FIRST_NAME)
  VALUES (’2112’, ’LIFESON’, ’ALEX’)
```

The PASSWORD, CIPHER and USING/GIVING NUMBER clauses cannot be used.

UPDATE

The Natural DML UPDATE statement updates a row in a DB2 table which has been read with a preceding FIND, READ, or SELECT statement. It corresponds to the SQL statement UPDATE WHERE CURRENT OF *cursor-name* (Positioned UPDATE), which means that only the row which was read last can be updated.

UPDATE when using the File Server

If a row rolled out to the file server is to be updated, Natural automatically rereads the original row from the database to compare it with its image stored in the file server. If the original row has not been modified in the meantime, the UPDATE operation is performed. With the next terminal I/O, the transaction is terminated and the row is definitely updated on the database.

If the UPDATE operates on a scrollable cursor, the row on the file server and the row in the base table are updated. If the row no longer qualifies for the search criteria of the related SELECT statement after the update, the row is marked as UPDATE hole on the file server.

However, if any modification is detected, the row will not be updated and Natural issues the NAT3703 error message for non-scrollable cursors.

If the UPDATE operates on a scrollable cursor, NDB simulates SQLCODE -224 (THE RESULT TABLE DOES NOT AGREE WITH THE BASE TABLE USING) for DB2 compliance.

If the UPDATE operates on a scrollable cursor and the row has become a hole, NDB simulates SQLCODE Œ222 (AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST A HOLE).

Since an UPDATE statement requires rereading a single row by Natural, a unique index must be available for this table. All columns which comprise the unique index must be part of the corresponding Natural view.

UPDATE with FIND/READ

As explained with the FIND statement, Natural translates a FIND statement into an SQL SELECT statement. When a Natural program contains a DML UPDATE statement, this statement is translated into an SQL UPDATE statement and a FOR UPDATE OF clause is added to the SELECT statement.

Example:

```
FIND EMPLOYEES WITH SALARY < 5000
  ASSIGN SALARY = 6000
  UPDATE
```

Natural would translate the above Natural statements into SQL and assign a cursor name (for example, CURSOR1) as follows:

```
DECLARE CURSOR1 CURSOR FOR
SELECT SALARY FROM EMPLOYEES WHERE SALARY < 5000
  FOR UPDATE OF SALARY
UPDATE EMPLOYEES SET SALARY = 6000
  WHERE CURRENT OF CURSOR1
```

Both the SELECT and the UPDATE statement refer to the same cursor.

Due to DB2 logic, a column (field) can only be updated if it is contained in the FOR UPDATE OF clause; otherwise updating this column (field) is rejected. Natural includes automatically all columns (fields) into the FOR UPDATE OF clause which have been modified anywhere in the Natural program or which are input fields as part of a Natural map.

However, a DB2 column is not updated if the column (field) is marked as "not updateable" in the Natural DDM. Such columns (fields) are removed from the FOR UPDATE OF list without any warning or error message. The columns (fields) contained in the FOR UPDATE OF list can be checked with the LISTSQL command.

The Adabas short name in the Natural DDM determines whether a column (field) can be updated.

The following table shows the ranges that apply:

Short-Name Range	Type of Field
AA - N9	non-key field that can be updated
Aa - Nz	non-key field that can be updated
OA - O9	primary key field
PA - P9	ascending key field that can be updated
QA - Q9	descending key field that can be updated
RA - X9	non-key field that cannot be updated
Ra - Xz	non-key field that cannot be updated
YA - Y9	ascending key field that cannot be updated
ZA - Z9	descending key field that cannot be updated
1A - 9Z	non-key field that cannot be updated
1a - 9z	non-key field that cannot be updated

Be aware that a primary key field is never part of a FOR UPDATE OF list. A primary key field can only be updated by using a non-cursor UPDATE operation (see also UPDATE in the section Natural SQL Statements).

A row read with a FIND statement that contains a SORTED BY clause cannot be updated (due to DB2 limitations as explained with the FIND statement). A row read with a READ LOGICAL cannot be updated either (as explained with the READ statement).

If a column is to be updated which is redefined as an array, it is strongly recommended to update the whole column and not individual occurrences; otherwise, results are not predictable. To do so, in reporting mode you can use the OBTAIN statement (as described in the Natural Statements documentation), which must be applied to all field occurrences in the column to be updated. In structured mode, however, all these occurrences must be defined in the corresponding Natural view.

The data locked by an UPDATE statement are released when an END TRANSACTION (COMMIT WORK) or BACKOUT TRANSACTION (ROLLBACK WORK) statement is executed by the program.

Note:
If a length indicator field or NULL indicator field is updated in a Natural program without updating the field (column) it refers to, the update of the column is not generated for DB2 and thus no updating takes place.

UPDATE with SELECT

In general, the DML UPDATE statement can be used in both structured and reporting mode. However, after a SELECT statement, only the syntax defined for Natural structured mode is allowed:

UPDATE [RECORD] [IN] [STATEMENT] [(*r* **)]**

This is due to the fact that in combination with the SELECT statement, the DML UPDATE statement is only allowed in the special case of:

```
   ...
SELECT ...
   INTO VIEW view-name
   ...
```

Thus, only a whole Natural view can be updated; individual columns (fields) cannot.

Example:

```
DEFINE DATA LOCAL
01 PERS VIEW OF SQL-PERSONNEL
  02 NAME
  02 AGE
END-DEFINE

SELECT *
  INTO VIEW PERS
  FROM SQL-PERSONNEL
  WHERE NAME LIKE 'S%'

    IF NAME = 'SMITH'
      ADD 1 TO AGE
    UPDATE
    END-IF

END-SELECT
    ...
```


In combination with the DML UPDATE statement, any other form of the SELECT statement is rejected and an error message is returned.

In all other respects, the DML UPDATE statement can be used with the SELECT statement in the same way as with the Natural FIND statement described earlier in this section and in the Natural Statements documentation.

Natural SQL Statements - Overview

This section covers points you have to consider when using Natural SQL statements with DB2. These DB2-specific points partly consist in syntax enhancements which belong to the Extended Set of Natural SQL syntax. The Extended Set is provided in addition to the Common Set to support database-specific features.

This section covers the following topics:

- Syntactical Items
- CALLDBPROC
- COMMIT
- DELETE
- INSERT
- PROCESS SQL
- READ RESULT SET
- ROLLBACK
- SELECT (cursor-oriented)
- SELECT SINGLE (non-cursor-oriented)
- UPDATE

Natural SQL Statements - Syntactical Items

The following common syntactical items are either DB2-specific and do not conform to the standard SQL syntax definitions (that is, to the Common Set of Natural SQL syntax) or impose restrictions when used with DB2 (see also SQL Statements in the Natural Statements documentation).

This section covers the following topics:

- atom
- comparison
- factor
- scalar-function
- column-function
- scalar-operator
- special-register
- units
- case-expression

---

atom

An atom can be either a parameter (that is, a Natural program variable or host variable) or a constant. When running dynamically, however, the use of host variables is restricted by DB2. For further details, refer to the relevant DB2 literature by IBM.

comparison

The comparison operators specific to DB2 belong to the Natural Extended Set. For a description, refer to Comparison Predicate in Search Conditions, Natural SQL Statements (Statements Grouped by Functions, Natural Statements documentation).

factor

The following three factors are specific to DB2 and belong to the Natural Extended Set:

special-register
scalar-function (*scalar-expression*, ...)
scalar-expression unit
case-expression

scalar-function

A scalar function is a built-in function that can be used in the construction of scalar computational expressions. Scalar functions are specific to DB2 and belong to the Natural Extended Set.

The scalar functions NDB supports are listed below in alphabetical order:

A - H	I - R	S - Z
ABS	IDENTITY_VAL_LOCAL	SECOND
ABSVAL	IFNULL	SIGN
ACOS	INSERT	SIN
ADD_MONTHS	INTEGER	SINH
ASIN	JULIAN_DAY	SMALLINT
ATAN	LAST_DAY	SPACE
ATAN2	LCASE	SQRT
ATANH	LEFT	STRIP
BLOB	LENGTH	SUBSTR
CCSID_ENCODING	LN	TAN
CEIL	LOCATE	TANH
CEILING	LOG	TIME
CHAR	LOG10	TIMESTAMP
CLOB	LOWER	TIMESTAMP_FORMAT
COALESCE	LTRIM	TO_CHAR
CONCAT	MAX	TO_DATE
COS	MICROSECOND	TRANSLATE
COSH	MIDNIGHT_SECONDS	TRUNC
DATE	MIN	TRUNC_TIMESTAMP
DAY	MINUTE	TRUNCATE
DAYOFMONTH	MOD	UCASE
DAYOFWEEK	MONTH	UPPER
DAYOFWEEK_ISO	MULTIPLY_ALT	VALUE
DAYOFYEAR	NEXT_DAY	VARCHAR
DAYS	NULLIF	VARCHAR_FORMAT
DBCLOB	POSSTR	VARGRAPHIC
DEC	POWER	WEEK
DECIMAL	QUARTER	WEEK_ISO
DEGREES	RADIANS	YEAR
DIGITS	RAISE_ERROR	
DOUBLE	RAND	
DOUBLE_PRECISION	REAL	
EXP	REPEAT	
FLOAT	REPLACE	
FLOOR	RIGHT	
GRAPHIC	ROUND	
HEX	ROUND_TIMESTAMP	
HOUR	ROWID	
	RTRIM	

Each scalar function is followed by one or more scalar expressions in parentheses. The number of scalar expressions depends upon the scalar function. Multiple scalar expressions must be separated from one another by commas.

Example:

```
SELECT NAME
  INTO NAME
  FROM SQL-PERSONNEL
  WHERE SUBSTR ( NAME, 1, 3 ) = 'Fri'
       ...
```


column-function

A column function returns a single-value result for the argument it receives. The argument is a set of like values, such as the values of a column. Column functions are also called aggregating functions.

The following column functions conform to standard SQL. They are not specific to DB2:

AVG
COUNT
MAX
MIN
SUM

The following column functions do not conform to standard SQL. They are specific to DB2 and belong to the Natural Extended Set.

COUNT_BIG
STDDEV
STDDEV_POP
STDDEV_SAMP
VAR
VAR_POP
VAR_SAMP
VARIANCE
VARIANCE_SAMP

scalar-operator

The concatenation operator (CONCAT or "||") does not conform to standard SQL. It is specific to DB2 and belongs to the Natural Extended Set.

special-register

The following special registers do not conform to standard SQL. They are specific to DB2 and belong to the Natural Extended Set:

CURRENT APPLICATION ENCODING SCHEME
CURRENT DATE
CURRENT_DATE
CURRENT DEGREE
CURRENT FUNCTION PATH
CURRENT_LC_CTYPE
CURRENT LC_CTYPE
CURRENT LOCALE LC_CTYPE
CURRENT OPTIMIZATION HINT
CURRENT PACKAGESET
CURRENT_PATH
CURRENT PRECISION
CURRENT RULES
CURRENT SQLID
CURRENT SERVER
CURRENT TIME
CURRENT_TIME
CURRENT TIMESTAMP
CURRENT TIMEZONE

CURRENT_TIMEZONE
USER

A reference to a special register returns a scalar value.

Using the command SET CURRENT SQLID, the creator name of a table can be substituted by the current SQLID. This enables you to access identical tables with the same table name but with different creator names.

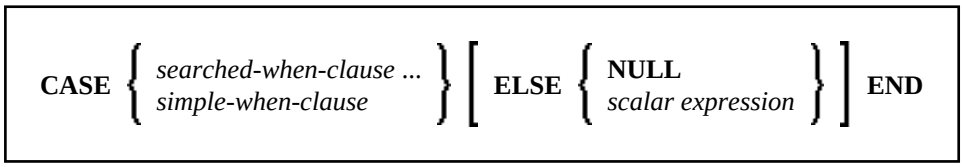
units

Units, also called durations, are specific to DB2 and belong to the Natural Extended Set.

The following units are supported:

DAY
DAYS
HOUR
HOURS
MICROSECOND
MICROSECONDS
MINUTE
MINUTES
MONTH
MONTHS
SECOND
SECONDS
YEAR
YEARS

case-expression



Case-expressions do not conform to standard SQL and are therefore supported by the Natural SQL Extended Set only.

Example:

```
DEFINE DATA LOCAL
01 #EMP
02 #EMPNO (A10)
02 #FIRSTNME (A15)
02 #MIDINIT  (A5)
02 #LASTNAME (A15)
02 #EDLEVEL   (A13)
02 #INCOME (P7)
END-DEFINE
SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME,
       (CASE WHEN EDLEVEL < 15 THEN 'SECONDARY'
             WHEN EDLEVEL < 19 THEN 'COLLEGE'
             ELSE             'POST GRADUATE'
        END ) AS EDUCATION, SALARY + COMM AS INCOME
       INTO
```



```
        #EMPNO, #FIRSTNME, #MIDINIT, #LASTNAME,
        #EDLEVEL, #INCOME
          FROM DSN8510-EMP
          WHERE (CASE WHEN SALARY = 0 THEN NULL
                                      ELSE SALARY / COMM
                                      END ) > 0.25
  DISPLAY #EMP
  END-SELECT
  END
```

NDB - CALLDBPROC

The CALLDBPROC statement is used to call DB2 stored procedures. It supports the result set mechanism of DB2 and it enables you to call DB2 stored procedures.

This section covers the following topics:

- Static and Dynamic Execution
- Result Sets
- List of Parameter Data Types
- CALLMODE=NATURAL
- Example of CALLDBPROC/READ RESULT SET

Futher details and syntax:
CALLDBPROC in Natural SQL Statements in the Natural Statements documentation.

Static and Dynamic Execution

If the CALLDBPROC statement is executed dynamically, all parameters and constants are mapped to the variables of the following DB2 SQL statement:

```
CALL :hv USING  DESCRIPTOR :sqlda  statement
```

:hv denotes a host variable containing the name of the procedure to be called and *:sqlda* is a dynamically generated sqlda describing the parameters to be passed to the stored procedure.

If the CALLDBPROC statement is executed statically, the constants of the CALLDBPROC statement are also generated as constants in the generated assembler SQL source for the DB2 precompiler.

Result Sets

If the SQLCODE created by the CALL statement indicates that there are result sets (SQLCODE +466 and +464), Natural for DB2 runtime executes a

```
DESCRIBE PROCEDURE :hv INTO :sqlda
```

statement in order to retrieve the result set locator values of the result sets created by the invoked stored procedure. These values are put into the RESULT SETS variables specified in the CALLDBPROC statement. Each RESULT SETS variable specified in a CALLDBPROC for which no result set locator value is present is reset to zero. The result set locator values can be used to read the result sets by means of the READ RESULT SET statement as long as the database transaction which created the result set has not yet issued a COMMIT or ROLLBACK.

If the result set was created by a cursor WITH HOLD, the result set locator value remains valid after a COMMIT operation.

Unlike other Natural SQL statements, CALLDBPROC enables you (optionally!) to specify a SQLCODE variable following the GIVING keyword which will contain the SQLCODE of the underlying CALL statement. If GIVING is specified, it is up to the Natural program to react on the SQLCODE (error message NAT3700 is not issued by the runtime).

List of Parameter Data Types

Below are the parameter data types supported by the CALLDBPROC statement:

Natural Format/Length	DB2 Data Type
A*n*	CHAR(*n*)
B2	SMALLINT
B4	INT
B*n* (*n* = not equal 2 or 4)	CHAR(*n*)
F4	REAL
F8	DOUBLE PRECISION
I2	SMALLINT
I4	INT
N*nn.m*	NUMERIC(*nn+m,m*)
P*nn.m*	NUMERIC(*nn+m,n*)
G*n*	GRAPHIC(*n*)
A*n*/1:m	VARCHAR(*n*m*)
D	DATE
T	TIME (see also TIME below)

TIME

The Natural format **T** has a wider data range than the equivalent DB2 TIME data type. Compared with DB2 TIME, in addition, the Natural **T** variable has a date fraction (year, month, day) and the tenths of a second.

As a result, converting a Natural **T** variable into a DB2 TIME value, NDB cuts off the date fraction and the tenths of a second part. Converting DB2 TIME into Natural **T**, the date fraction is reset to 0000-01-02 and the tenths of a second part is reset to 0 in Natural.

CALLMODE=NATURAL

This parameter is used to invoke Natural stored procedures defined with PARAMETER STYLE GENERAL/WITH NULL.

If the CALLMODE=NATURAL parameter is specified, an additional parameter describing the parameters passed to the Natural stored procedure is passed from the client, i.e. caller, to the server, i.e. the NDB server stub. The parameter is the Stored Procedure Control Block (STCB; see also STCB Layout in PARAMETER STYLE in the section Natural Stored Procedures and UDFs) and has the format VARCHAR from the viewpoint of DB2. Therefore, every Natural stored procedure defined with PARAMETER STYLE GENERAL/WITH NULL has to be defined in the SYSIBM.SYSPROCEDURES table (only applies to DB2 for OS/390 Version 5 and below) or with the CREATE PROCEDURE statement (DB2 UDB for OS/390 Version 6 and above) by using this VARCHAR parameter as the first in its PARMLIST row.

From the viewpoint of the caller, i.e. the Natural program, and from the viewpoint of the stored procedure, i.e. Natural subprogram, the STCB is invisible. It is passed as first parameter by the Natural for DB2 runtime and it is used as on the server side to build the copy of the passed data in the Natural thread and the corresponding CALLNAT statement. Additionally, this parameter serves as a container for error information created during execution of the Natural stored procedure by the Natural runtime. It also contains information on the library where you are logged on and the Natural subprogram to be invoked.

Example of CALLDBPROC/READ RESULT SET

Below is an example program for issuing CALLDBPROC and READ RESULT SET statements:

```
DEFINE DATA LOCAL
1 ALPHA       (A8)
1 NUMERIC     (N7.3)
1 PACKED      (P9.4)
1 VCHAR       (A20/1:5) INIT     <'DB25SGCP'>
1 INTEGER2    (I2)
1 INTEGER4    (I4)
1 BINARY2     (B2)
1 BINARY4     (B4)
1 BINARY12    (B12)
1 FLOAT4      (F4)
1 FLOAT8      (F8)
1 INDEX-ARRAY (I2/1:11)
1 INDEX-ARRAY1(I2)
1 INDEX-ARRAY2(I2)
1 INDEX-ARRAY3(I2)
1 INDEX-ARRAY4(I2)
1 INDEX-ARRAY5(I2)
1 INDEX-ARRAY6(I2)
1 INDEX-ARRAY7(I2)
1 INDEX-ARRAY8(I2)
1 INDEX-ARRAY9(I2)
1 INDEX-ARRAY10(I2)
1 INDEX-ARRAY11(I2)
1 #RESP        (I4)
1 #RS1         (I4) INIT <99>
1 #RS2         (I4) INIT <99>
LOCAL
1 V1 VIEW OF SYSIBM-SYSTABLES
2 NAME
1 V2 VIEW OF SYSIBM-SYSPROCEDURES
2 PROCEDURE
2 RESULT_SETS
1 V (I2) INIT <99>
END-DEFINE
CALLDBPROC 'DAEFDB25.SYSPROC.SNGSTPC'  DSN8510-EMP
 ALPHA  INDICATOR :INDEX-ARRAY1
 NUMERIC INDICATOR :INDEX-ARRAY2
 PACKED  INDICATOR :INDEX-ARRAY3
 VCHAR(*) INDICATOR :INDEX-ARRAY4
 INTEGER2 INDICATOR :INDEX-ARRAY5
 INTEGER4 INDICATOR :INDEX-ARRAY6
 BINARY2  INDICATOR :INDEX-ARRAY7
 BINARY4  INDICATOR :INDEX-ARRAY8
 BINARY12 INDICATOR :INDEX-ARRAY9
 FLOAT4   INDICATOR :INDEX-ARRAY10
 FLOAT8   INDICATOR :INDEX-ARRAY11
  RESULT SETS #RS1 #RS2
 CALLMODE=NATURAL
```



```
READ (10) RESULT SET #RS2 INTO VIEW V2 FROM SYSIBM-SYSTABLES
WRITE 'PROC F RS  :' PROCEDURE 50T RESULT_SETS
END-RESULT
END
```


NDB - COMMIT

Futher details and syntax:
COMMIT in Natural SQL Statements in the Natural Statements documentation.

The SQL COMMIT statement indicates the end of a logical transaction and releases all DB2 data locked during the transaction. All data modifications are made permanent.

COMMIT is a synonym for the Natural END TRANSACTION statement as described in the section Natural DML Statements.

No transaction data can be provided with the COMMIT statement.

If this command is executed from a Natural stored procedure or user-defined function (UDF), Natural for DB2 does not execute the underlying commit operation. This allows the Natural stored procedure or UDF to commit updates against non DB2 databases.

Under CICS, the COMMIT statement is translated into an EXEC CICS SYNCPOINT command. If the file server is used, an implicit end-of-transaction is issued after each terminal I/O. This is due to CICS-specific transaction processing in pseudo-conversational mode.

Under IMS/TM, the COMMIT statement is not translated into an IMS Checkpoint command, but is ignored. An implicit end-of-transaction is issued after each terminal I/O. This is due to IMS/TM-specific transaction processing.

Unless when used in combination with the WITH HOLD clause (see SELECT Cursor-oriented), a COMMIT statement must not be placed within a database loop, since all cursors are closed when a logical unit of work ends. Instead, it has to be placed outside such a loop or after the outermost loop of nested loops.

If an external program written in another standard programming language is called from a Natural program, this external program must not contain its own COMMIT command if the Natural program issues database calls, too. The calling Natural program must issue the COMMIT statement for the external program.

NDB - DELETE - SQL

Both the cursor-oriented or Positioned DELETE, and the non-cursor or Searched DELETE SQL statements are supported as part of Natural SQL; the functionality of the Positioned DELETE statement corresponds to that of the Natural DML DELETE statement.

With DB2, a table name in the FROM clause of a Searched DELETE statement can be assigned a *correlation-name*. This does not correspond to the standard SQL syntax definition and therefore belongs to the Natural Extended Set.

The Searched DELETE statement must be used, for example, to delete a row from a self-referencing table, since with self-referencing tables a Positioned DELETE is not allowed by DB2.

Futher details and syntax:
DELETE in Natural SQL Statements in the Natural Statements documentation.

NDB - INSERT

The INSERT statement is used to add one or more new rows to a table.

Since the INSERT statement can contain a select expression, all the DB2-specific syntactical items described above apply.

Futher details and syntax:
INSERT in Natural SQL Statements in the Natural Statements documentation.

NDB - PROCESS SQL

The PROCESS SQL statement is used to issue SQL statements to the underlying database. The statements are specified in a *statement-string*, which can also include constants and parameters.

The set of statements which can be issued is also referred to as Flexible SQL and comprises those statements which can be issued with the SQL statement EXECUTE.

In addition, Flexible SQL includes the following DB2-specific statements:

CALL
CONNECT
SET APPLICATION ENCODING SCHEME
SET CONNECTION
SET CURRENT DEGREE
SET CURRENT LC_CTYPE
SET CURRENT OPTIMIZATION HINT
SET CURRENT PACKAGESET
SET CURRENT PATH
SET CURRENT PRECISION
SET CURRENT RULES
SET CURRENT SQLID
SET *host-variable= special-register* RELEASE

Note:
To avoid transaction synchronization problems between the Natural environment and DB2, the COMMIT and ROLLBACK statements must not be used within PROCESS SQL.

Futher details and syntax:
PROCESS SQL in Natural SQL Statements in the Natural Statements documentation.

NDB - READ RESULT SET

The READ RESULT SET statement reads a result set created by a Natural stored procedure that was invoked by a CALLDBPROC statement (see the relevant section).

For details on how to specify the scroll direction by using the variable *scroll-hv*, see the SELECT statement in the section Natural SQL Statements.

Futher details and syntax:
READ RESULT SET in Natural SQL Statements in the Natural Statements documentation.

NDB - ROLLBACK

Futher details and syntax:
ROLLBACK in Natural SQL Statements in the Natural Statements documentation.

The SQL ROLLBACK statement undoes all database modifications made since the beginning of the last logical transaction. Logical transactions can start either after the beginning of a session or after the last COMMIT/END TRANSACTION or ROLLBACK/BACKOUT TRANSACTION statement. All records held during the transaction are released.

ROLLBACK is a synonym for the Natural statement BACKOUT TRANSACTION as described in the section Natural DML Statements.

If this command is executed from a Natural stored procedure or user-defined function (UDF), Natural for DB2 executes the underlying rollback operation. This sets the caller into a must-rollback state. If this command is executed by Natural error processing (implicit ROLLBACK), Natural for DB2 does not execute the underlying rollback operation, thus allowing the caller to receive the original Natural error.

Under CICS, the ROLLBACK statement is translated into an EXEC CICS ROLLBACK command. However, if the file server is used, only changes made to the database since the last terminal I/O are undone. This is due to CICS-specific transaction processing in pseudo-conversational mode.

Under IMS/TM, the ROLLBACK statement is translated into an IMS Rollback (ROLB) command. However, only changes made to the database since the last terminal I/O are undone. This is due to IMS/TM-specific transaction processing.

As all cursors are closed when a logical unit of work ends, a ROLLBACK statement must not be placed within a database loop; instead, it has to be placed outside such a loop or after the outermost loop of nested loops.

If an external program written in another standard programming language is called from a Natural program, this external program must not contain its own ROLLBACK command if the Natural program issues database calls, too. The calling Natural program must issue the ROLLBACK statement for the external program.

NDB - SELECT - Cursor-Oriented

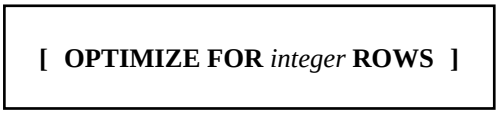
Like the Natural FIND statement, the cursor-oriented SELECT statement is used to select a set of rows (records) from one or more DB2 tables, based on a search criterion. Since a database loop is initiated, the loop must be closed by a LOOP (reporting mode) or END-SELECT statement. With this construction, Natural uses the same loop processing as with the FIND statement.

In addition, no cursor management is required from the application program; it is automatically handled by Natural.

Below is information on:

- OPTIMIZE FOR integer ROWS
- WITH - Isolation Level
- QUERYNO
- FETCH FIRST
- WITH HOLD
- WITH RETURN
- WITH INSENSITIVE/SENSITIVE

OPTIMIZE FOR integer ROWS



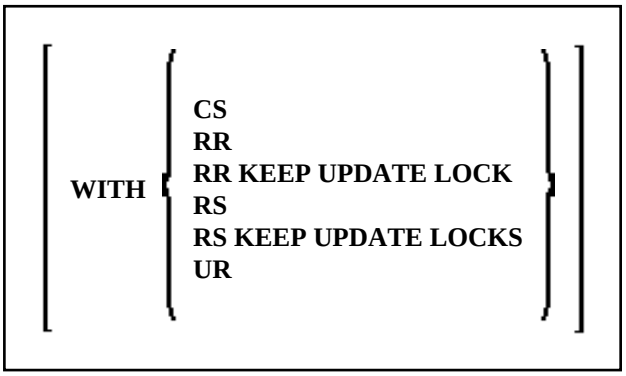
The OPTIMIZE FOR *integer* ROWS clause is used to inform DB2 in advance of the number (*integer*) of rows to be retrieved from the result table. Without this clause, DB2 assumes that all rows of the result table are to be retrieved and optimizes accordingly.

This optional clause is useful if you know how many rows are likely to be selected, because optimizing for *integer* rows can improve performance if the number of rows actually selected does not exceed the *integer* value (which can be in the range from 0 to 2147483647).

Example:

```
SELECT name INTO #name FROM table
WHERE AGE = 2 OPTIMIZE FOR 100 ROWS
```

WITH - Isolation Level



This WITH clause allows you to specify an explicit isolation level with which the statement is to be executed. The following options are provided:

Option	Meaning
CS	Cursor stability
RR	Repeatable Read
RS	Read Stability
RS KEEP UPDATE LOCKS	Only valid if a FOR UPDATE OF clause is specified. Read Stability and retaining update locks.
RR KEEP UPDATE LOCKS	Only valid if a FOR UPDATE OF clause is specified. Repeatable Read and retaining update locks.
UR	Uncommitted Read

WITH UR can only be specified within a SELECT statement and when the table is read-only. The default isolation level is determined by the isolation of the package or plan into which the statement is bound. The default isolation level also depends on whether the result table is read-only or not. To find out the default isolation level, refer to the IBM literature.

Note:
This option also works for non-cursor selection.

QUERYNO

[QUERYNO *integer* **]**

The QUERNO clause specifies the number to be used for this SQL statement in EXPLAIN output and trace records. The number is used as QUERYNO column in the PLAN_TABLE for the rows that contain information on this statement.

FETCH FIRST

[FETCH FIRST { **1** / *integer* } { **ROWS** / **ROW** } **ONLY]**

The FETCH FIRST clause limits the number of rows to be fetched. It improves the performance of queries with potentially large result sets if only a limited number of rows is needed.

WITH HOLD

[WITH HOLD]

The WITH HOLD clause is used to prevent cursors from being closed by a commit operation within database loops. If WITH HOLD is specified, a commit operation commits all the modifications of the current logical unit of work, but releases only locks that are not required to maintain the cursor. This optional clause is mainly useful in batch mode; it is ignored in CICS pseudo-conversational mode and in IMS message-driven programs.

Example:

```
SELECT name INTO #name FROM table
WHERE AGE = 2 WITH HOLD
```

WITH RETURN

[WITH RETURN]

The WITH RETURN clause is used to create result sets. Therefore, this clause only applies to programs which operate as Natural stored procedure. If the WITH RETURN clause is specified in a SELECT statement, the underlying cursor remains open when the associated processing loop is left, except when the processing loop had read all rows of the result set itself. During first execution of the processing loop, only the cursor is opened. The first row is not yet fetched. This allows the Natural program to return a full result set to the caller of the stored procedure. It is up to the Natural you to decide how many rows are processed by the Natural stored procedure and how many unprocessed rows of the result set are returned to the caller of the stored procedure. If you want to process rows of the select operation in the Natural stored procedure, you must define

```
IF *counter =1 ESCAPE TOP END-IF
```

in order to avoid processing of the first "empty row" in the processing loop. If you decide to terminate the processing of rows, you must define

```
If condition ESCAPE BOTTOM END-IF
```

in the processing loop.

If the program reads all rows of the result set, the cursor is closed and no result set is returned for this SELECT WITH RETURN to the caller of the stored procedure.

The following programs are examples for retrieving full result sets (Example 1) and partial result sets (Example 2).

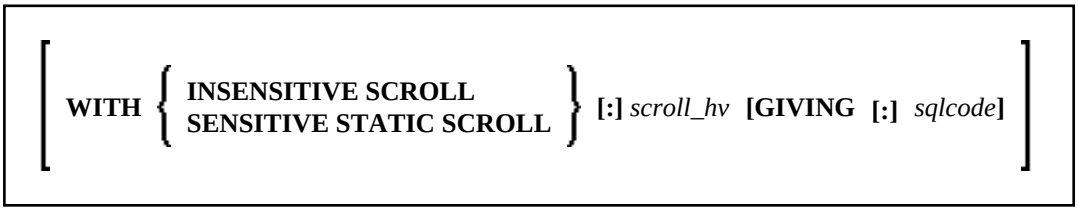
Example 1:

```
DEFINE DATA LOCAL
. . .
END DEFINE
*
*  Return all rows of the result set
*
SELECT * INTO VIEW V2
                    FROM SYSIBM-SYSROUTINES
                    WHERE RESULT_SETS > 0
                    WITH RETURN
ESCAPE BOTTOM
END-SELECT
END
```

Example 2:

```
DEFINE DATA LOCAL
. . .
END DEFINE
*
*  Read the first two rows and return the rest as result set
*
SELECT * INTO VIEW V2
                    FROM SYSIBM-SYSROUTINES
                    WHERE RESULT_SETS > 0
                    WITH   RETURN
WRITE PROCEDURE *COUNTER
IF *COUNTER = 1 ESCAPE TOP END-IF
IF *COUNTER =3 ESCAPE BOTTOM END-IF
END-SELECT
END
```

WITH INSENSITIVE/SENSITIVE



NDB supports DB2 scrollable cursors by using the clauses WITH INSENSITIVE SCROLL and WITH SENSITIVE STATIC SCROLL. Scrollable cursors allow NDB applications to position randomly any row in a result set. With non-scrollable cursors, the data can only be read sequentially, from top to bottom.

Scrollable cursors use temporary result tables and require a TEMP database in DB2 (see the relevant DB2 literature by IBM).

INSENSITIVE SCROLL refers to a cursor that cannot be used in Positioned UPDATE or Positioned DELETE operations. In addition, once opened, an INSENSITIVE SCROLL cursor does not reflect UPDATEs, DELETEs or INSERTs against the base table, after the cursor was opened.

SENSITIVE STATIC SCROLL refers to a cursor that can be used for Positioned UPDATEs or Positioned DELETE operations. In addition, a SENSITIVE STATIC SCROLL cursor reflects UPDATEs, DELETEs of base table rows. The cursor does not reflect INSERT operations.

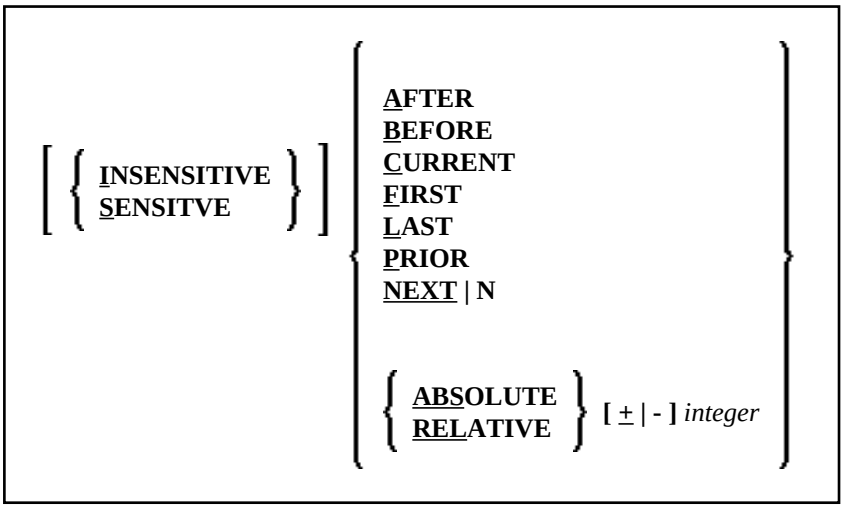
Below is information on:

- scroll_hv
- GIVING [:] sqlcode

scroll_hv

The variable scroll_hv must be alphanumeric.

The variable *scroll_hv* specifies which row of the result table will be fetched during one execution of the database processing loop. Additionally, it specifies the sensitivity of UPDATEs or DELETEs against the base table row during a FETCH operation. The contents of *scroll_hv* is evaluated each time the database processing loop cycle is executed.



scroll_hv - Sensitivity Specification

The specification of the sensitivity INSENSITIVE or SENSITIVE is optional.

If it is omitted from a FETCH against an INSENSITIVE SCROLL cursor, the default will be INSENSITIVE.

If it is omitted from a FETCH against a SENSITIVE STATIC SCROLL cursor, the

The sensitivity specifies whether or not the rows in the base table are checked when performing a FETCH operation for a scrollable cursor.

If the corresponding base table column qualifies for the WHERE clause and has not been deleted, a SENSITIVE FETCH will return the row of the base table.

If the corresponding base table column does not qualify for the WHERE clause or has not been deleted, a SENSITIVE FETCH will return an UPDATE hole or a DELETE hole state (SQLCODE +222).

An INSENSITIVE FETCH will not check the corresponding base table column.

scroll_hv - Options

Below is an explanation of the options available to determine the row(s) to fetch, the position from where to start the fetch and/or the direction in which to scroll:

Option	Explanation
AFTER	Positions after the last row. No row is fetched.
BEFORE	Positions before the first. No row is fetched.
CURRENT	Fetches the current row (again).
FIRST	Fetches the first row.
LAST	Fetches the last row.
NEXT	Fetches the row after the current one. This is the default value.
PRIOR	Fetch the row before the current one.
+/- *integer*	Only applies in connection with ABSOLUTE or RELATIVE. Specifies the position of the row to be fetched ABSOLUTE or RELATIVE. Enter a plus (+) or minus (-) sign followed by an integer. The default value is a plus (+).
ABSOLUTE	Only applies in connection with +/- *integer*. Uses *integer* as the absolute position within the result set from where the row is fetched. See the DB2 SQL reference by IBM about further details regarding positive and negative position numbers.
RELATIVE	Only applies in connection with +/- *integer*. Uses *integer* as the relative position to the current position within the result set from where the row is fetched. See the DB2 SQL reference by IBM about further details regarding positive and negative position numbers.

GIVING [:] sqlcode

The specification of GIVING [:] *sqlcode* is optional. If specified, the Natural variable [:] *sqlcode* must be of the Format I4. The values for this variable are returned from the DB2 SQLCODE of the underlying FETCH operation. This allows the application to react to different statuses encountered while the scrollable cursor is open. The most important status codes indicated by SQLCODE are listed in the following table:

SQLCODE	Explanation
0	FETCH operation successful, data returned except for FETCH with option BEFORE or AFTER.
+100	Row not found, cursor still open, no data returned.
+222	UPDATE or DELETE hole, cursor still open, no data returned. The corresponding row of the base table has been updated or deleted, so that the row no longer qualifies for the WHERE clause.
+231	Fetch operation with the option CURRENT, but cursor not positioned on any row, no data returned. This occurs if the previous FETCH returned SQLCODE +100.

If you specify GIVING [:] *sqlcode*, the application must react to the different statuses. If a SQLCODE +100 is entered five times successively and without terminal I/O, the NDB runtime will issue Natural Error NAT3296 in order to avoid application looping. The application can terminate the processing loop by executing an ESCAPE statement.

If you do not specify GIVING [:] *sqlcode*, except for SQLCODE 0 and SQLCODE +100, each SQLCODE will generate Natural Error NAT3700 and the processing loop will be terminated. SQLCODE +100 (row not found) will terminate the processing loop.

See also the example program DEM2SCRL supplied in the Natural system library SYSDB2.

SELECT SINGLE - Non-Cursor-Oriented

The Natural statement SELECT SINGLE provides the functionality of a non-cursor selection (singleton SELECT); that is, a select expression that retrieves at most one row without using a cursor.

Since DB2 supports the singleton SELECT command in static SQL only, in dynamic mode, the Natural SELECT SINGLE statement is executed like a set-level SELECT statement, which results in a cursor operation. However, Natural checks the number of rows returned by DB2. If more than one row is selected, a corresponding error message is returned.

Related Topic:
See also the SELECT statement for a cursor-oriented selection of rows.

NDB - UPDATE - SQL

Both the cursor-oriented or Positioned UPDATE, and the non-cursor or Searched UPDATE SQL statements are supported as part of Natural SQL. Both of them reference either a table or a Natural view.

With DB2, the name of a table or Natural view to be referenced by a Searched UPDATE can be assigned a *correlation-name*. This does not correspond to the standard SQL syntax definition and therefore belongs to the Natural Extended Set.

The Searched UPDATE statement must be used, for example, to update a primary key field, since DB2 does not allow updating of columns of a primary key by using a Positioned UPDATE statement.

> **Note:**
> If you use the SET * notation, all fields of the referenced Natural view are added to the FOR UPDATE OF and SET lists. Therefore, ensure that your view contains only fields which can be updated; otherwise, a negative SQLCODE is returned by DB2.

Futher details and syntax:
UPDATE in Natural SQL Statements in the Natural Statements documentation.

NDB - Natural System Variables

When used with DB2, there are restrictions for the following Natural system variables:

- *ISN
- *NUMBER

---

*ISN

As there is no DB2 equivalent of Adabas ISNs, the system variable *ISN is not applicable to DB2 tables.

*NUMBER

When used with a FIND NUMBER or HISTOGRAM statement, *NUMBER contains the number of rows actually found.

When applied to data from a DB2 table in any other case, the system variable *NUMBER only indicates whether any rows have been found. If no rows have been found, *NUMBER is "0". Any value other than "0" indicates that at least one row has been found; however, the value contained in *NUMBER has no relation to the number of rows actually found.

The reason is that if *NUMBER were to produce a valid number, Natural would have to translate the corresponding FIND statement into an SQL SELECT statement including the special function COUNT(*); however, a SELECT containing a COUNT function would produce a read-only result table, which would not be available for updating. In other words, the option to update selected data was given priority in Natural over obtaining the number of rows that meet the search criteria.

To obtain the number of rows affected by the Natural SQL statements Searched UPDATE, Searched DELETE and INSERT, the Natural subprogram NDBNROW is provided. Alternatively, you can use the Natural system variable *ROWCOUNT as described in the Natural System Variables documentation.

NDB - Error Handling

In contrast to the normal Natural error handling, where either an ON ERROR statement is used to intercept execution time errors or standard error message processing is performed and program execution is terminated, the enhanced error handling of Natural for DB2 provides an application controlled reaction to the encountered SQL error.

Two Natural subprograms, NDBERR and NDBNOERR, are provided to disable the usual Natural error handling and to check the encountered SQL error for the returned SQL code. This functionality replaces the "E" function of the DB2SERV interface, which is still provided but no longer documented.

For further information on Natural subprograms provided for DB2, see the section Natural Subprograms.

Example:

```
DEFINE DATA LOCAL
01 #SQLCODE             (I4)
01 #SQLSTATE            (A5)
01 #SQLCA               (A136)
01 #DBMS                (B1)
END-DEFINE
*
*        Ignore error from next statement
*
CALLNAT 'NDBNOERR'
*
*        This SQL statement produces an SQL error
*
INSERT INTO SYSIBH-SYSTABLES (CREATOR, NAME, COLCOUNT)
  VALUES ('SAG', 'MYTABLE', '3')
*
*        Investigate error
*
CALLNAT 'NDBERR' #SQLCODE #SQLSTATE #SQLCA #DBMS
*
IF #DBMS NE 2                              /* not DB2
  MOVE 3700 TO *ERROR-NR
END-IF
*
DECIDE ON FIRST VALUE OF #SQLCODE
  VALUE 0, 100                             /* successful execution
    IGNORE
  VALUE -803                               /* duplicate row
    /* UPDATE existing record
    /*
    IGNORE
  NONE VALUE
    MOVE 3700 TO *ERROR-NR
END-DECIDE
*
END
```


NDB - Natural Stored Procedures and UDFs

NDB supports the writing and executing of Natural stored procedures and Natural user-defined functions (Natural UDFs).

Natural stored procedures are user-written programs that are invoked by the SQL statement CALL and executed by DB2 in the SPAS (Stored Procedure Address Space). SPAS is a separate address space reserved for stored procedures.

A function is an operation denoted by a function name followed by zero or more operands that are enclosed in parentheses. A function represents a relationship between a set of input values and a set of result values. If a function has been implemented by a user-written program, DB2 refers to it as a user-defined function (UDF).

This section covers the following topics:

- Types of Natural UDF
- PARAMETER STYLE
- Writing a Natural Stored Procedure
- Writing a Natural UDF
- Example Stored Procedure
- Example UDF
- Security

NDB - Types of UDF

NDB - Types of UDF

There are two types of Natural UDF:

- scalar UDF
- table UDF

The scalar UDF accepts several input arguments and returns one output value. It can be invoked by any SQL statement like a DB2 built-in-function.

The table UDF accepts several input arguments and returns a set of output values comprising one table row during each invocation.

You invoke a table UDF with a SELECT statement by specifying the table-function name in the FROM clause. A table UDF performs as a DB2 table and is invoked for each FETCH operation for the table-function specified in the SELECT statement.

NDB - PARAMETER STYLE

The PARAMETER STYLE identifies the linkage convention used to pass parameters to a DB2 stored procedure or a DB2 UDF.

This section describes the PARAMETER STYLEs and the STCB NDB uses for processing Natural DB2 stored procedures or Natural UDFs. Note that PARAMETER STYLE GENERAL (or GENERAL WITH NULL) and STCB Layout only apply to Natural stored procedures.

- GENERAL and GENERAL WITH NULL
- STCB Layout
- DB2SQL

GENERAL and GENERAL WITH NULL

Only applies to Natural stored procedures.

A Natural stored procedure defined with PARAMETER STYLE GENERAL only receives the user parameters specified.

A Natural stored procedure defined with PARAMETER STYLE GENERAL WITH NULL receives the user parameters specified and, additionally, a NULL indicator array that contains one NULL indicator for each user parameter.

Natural stored procedures defined with PARAMETER STYLE GENERAL/WITH NULL, require that the definition of the stored procedure within the DB2 catalog includes one additional parameter of the type VARCHAR in front of the user parameters of the stored procedure.

This parameter in front of the parameters is the STCB (Stored Procedure Control Block); see also STCB Layout below.

Below is information on:

- STCB - Stored Procedure Control Block
- Example of PARAMETER STYLE GENERAL
- Example of GENERAL WITH NULL

STCB - Stored Procedure Control Block

The STCB contains information the NDB server stub uses to execute Natural stored procedures, such as the library and the subprogram to be invoked. It also contains the format descriptions of the parameters passed to the stored procedure.

The STCB is invisible to the Natural stored procedure called. The STCB is evaluated by the NDB server stub and stripped off the parameter list that is passed to the Natural stored procedure.

If the caller of a Natural stored procedure defined with PARAMETER STYLE GENERAL/WITH NULL is a Natural program, the program must use a CALLDBPROC statement with the keyword CALLMODE=NATURAL.

If the caller of the Natural stored procedure is **not** a Natural program, the caller has to set up the STCB for the DB2 CALL statement and pass the STCB as the first parameter.

If an error occurs during the execution of a Natural stored procedure defined with PARAMETER STYLE GENERAL/WITH NULL, the error message text is returned to the STCB.

If the caller is a Natural program that uses CALLDBPROC and CALLMODE=NATURAL, the NDB runtime will wrap up the error text in the NAT3286 error message.

Example of PARAMETER STYLE GENERAL

In the Natural stored procedure, define the parameters as shown in the example program below:

Example of PARAMETER STYLE GENERAL

```
DEFINE DATA PARAMETER
01 P1 ...
01 P2 ...
...
...
01 Pn ...
LOCAL
...
...
END-DEFINE
```

Example of GENERAL WITH NULL

In the Natural stored procedure, define the parameters as shown in the example program below:

Example of PARAMETER STYLE GENERAL WITH NULL

```
DEFINE DATA PARAMETER
01 P1 ...
01 P2 ...
...
...
01 Pn ...
01 NULL-INDICATOR-ARRAY  (I2/1:n)
LOCAL
...
...
END-DEFINE
```

STCB Layout

Only applies to Natural stored procedures.

The following table describes the first parameter passed between the caller and the Natural stored procedure if CALLMODE = NATURAL (see also the relevant section in CALLDBPROC) is specified.

NAME	FORMAT	PROCESSING MODE SERVER
STCBL	I2	Input (size of following information)
Procedure Information		
STCBLENG	A4	Input (printable STCBL)
STCBID	A4	Input (’STCB’)
STCBVERS	A4	Input (version of STCB ’310 ’)
STCBUSER	A8	Input (user ID)
STCBLIB	A8	Input (library)
STCBPROG	A8	Input (calling program)
STCBPSW	A8	Unused (password)
STCBSTNR	A4	Input (CALLDBPROC statement number)
STCBSTPC	A8	Input (procedure called)
STCBPANR	A4	Input (number of parameters)
Error Information		
STCBERNR	A5	Output (Natural error number)
STCBSTAT	A1	Unused (Natural error status)
STCBLIB	A8	Unused (Natural error library)
STCBPRG	A8	Unused (Natural error program)
STCBLVL	A1	Unused (Natural error level)
STCBOTP	A1	Unused (error object type)
STCBEDYL	A2	Output (error text length)
STCBEDYT	A88	Output (error text)
	A100	Reserved for future use
Parameter Information		
STCBPADE	A variable	Input. See also PARAMETER DESCRIPTION (STCBPADE) below.

Below is information on:

- PARAMETER DESCRIPTION (STCBPADE)

PARAMETER DESCRIPTION (STCBPADE)

PARAMETER DESCRIPTION contains a description for each parameter passed to the Natural stored procedure consisting of parameter type, format specification and length. Parameter type is the AD attribute of the CALLNAT statement as described in the Natural Statements documentation.

Each parameter has the following format description element in the STCBPADE string

atl,*p*[,*d1*]....

where

- *a* is an attribute mark which specifies the parameter type:

Mark	Type	Equivalent AD Attribute	Equivalent DB2 Clause
M	modifiable	AD=M	INOUT
O	non-modifiable	AD=O	IN
A	input only	AD=A	OUT

- *t* is one of the following Natural format tokens:

t	Description	*l*	*p*	*dl*	Example
A	Alphanumeric	1-253	0	1-32767 or -	A30,0 or A30,0,10
N	Numeric unpacked	1-29	0-7	-	N10,3
P	Packed numeric	1-29	0-7	-	P13,4
I	Integer	2 or 4	0	-	I2,0
F	Floating point		0	-	I4,0
B	Binary		0	-	B23,0
D	Date	6	0	-	D6
T	Time	12	0	-	T12
L	Logical (unsupported)				

- *l* is an integer denoting the length/scale of the field. For numeric and packed numeric fields, *l* denotes the total number of digits of the field that is, the sum of the digits left and right of the decimal point. The Natural format N7.3 is, for example, represented by N10.3. See also the table above.
- *p* is an integer denoting the precision of the field. It is usually 0, except for numeric and packed fields where it denotes the number of digits right of the decimal point. See also the table above.
- *d1* is also an integer denoting the occurrences of the alphanumeric array (alphanumeric only). See also the table above.

This descriptive/control parameter is invisible to the calling Natural program and to the called Natural stored procedure, but it has to be defined in the parameter definition of the stored procedure row in the SYSIBM.SYSPROCEDURES table (only applies to DB2 for OS/390 Version 5 and below) or with the CREATE PROCEDURE statement (DB2 UDB for OS/390 Version 6 and above) and the DB2 PARAMETER STYLE GENERAL/WITH NULL.

The following table shows the number of parameters which have to be defined in the SYSIBM.SYSPROCEDURES table (only applies to DB2 for OS/390 Version 5 and below) or with the CREATE PROCEDURE statement (DB2 UDB for OS/390 Version 6 and above) for a Natural stored procedure defined with PARAMETER STYLE GENERAL depending on the number of user parameters and whether the client (i.e. the caller of a stored procedure for DB2 for OS/390) and the server (i.e. the stored procedure for DB2 for OS/390) is written in Natural or in another standard programming host language. *n* denotes the number of user parameters.

Client\Server	Natural	not Natural
Natural	*n* + 1	*n* (CALLMODE=NONE)
non-Natural	*n* + 1	*n*

DB2SQL

PARAMETER DB2SQL applies to Natural stored procedures and Natural UDFs.

A Natural stored procedure or Natural UDF with PARAMETER STYLE DB2SQL first receives the user parameters specified and then the parameters listed below, under Additional Parameters passed. For a Natural UDF, the input parameters are passed before the output parameters.

Additional Parameters passed:

- A NULL indicator for each user parameter of the CALL statement,
- The SQLSTATE to be returned to DB2,
- The qualified name of the Natural stored procedure or UDF,
- The specific name of the Natural stored procedure or UDF,
- The SQL DIAGNOSE field with a diagnostic string to be returned to DB2.

The SQLSTATE, the qualified name, the specific name and the DIAGNOSE field are defined in the Natural parameter data area (PDA) DB2SQL_P which is supplied in the Natural system library SYSDB2.

If the optional feature SCRATCHPAD *nnn* is specified additionally in the CREATE FUNCTION statement for the Natural UDF, the SCRATCHPAD storage parameter is passed to the Natural UDF.

Use the following definitions:

```
01 SCRATCHPAD A(4+nnn)
01 REDEFINE SCRATCHPAD
02 SCRATCHPAD_LENGTH(I4)
02 ...
```

Redefine the SCRATCHPAD in the Natural UDF according to your requirements.

The first four bytes of the SCRATCHPAD area contain an integer length field. Before initially invoking the Natural UDF with an SQL statement, DB2 resets the SCRATCHPAD area to x’00’ and sets the size *nnn* specified for the SCRATCHPAD into the first four bytes as an integer value.

Thereafter, DB2 does not reinitialize the SCRATCHPAD between the invocations of the Natural UDF for the invoking SQL statement. Instead, after returning from the Natural UDF, the contents of the SCRATCHPAD is preserved and restored at the next invocation of the Natural UDF.

Below is information on:

- Parameter CALL TYPE
- Parameter DBINFO
- Determining Library, Subprogram and Parameter Formats
- Invoking a Natural Stored Procedure
- Error Handling
- Lifetime of Natural Session
- Example of DB2SQL - Natural Stored Procedure
- Example of DB2SQL - Natural UDF

Parameter CALL TYPE

This parameter is optional and only applies to Natural UDFs.

The CALL TYPE parameter is passed if the FINAL CALL option is specified for a Natural scalar UDF, or if the Natural UDF is a table UDF. The CALL TYPE parameter is an integer indicating the type of call DB2 performs on the Natural UDF. See the DB2 SQL GUIDE for details on the parameter values provided in the CALL_TYPE parameter.

Parameter DBINFO

This parameter is optional.

If the option DBINFO is used, the DBINFO structure is passed to the Natural stored procedure or UDF. The DBINFO structure is described in the Natural PDA DBINFO_P supplied in the Natural system library SYSDB2.

Determining Library, Subprogram and Parameter Formats

The NDB server stub determines the subprogram and the library from the qualified and specific name of the Natural stored procedure or UDF. The SCHEMA name is used as library name, and the procedure or function name is used as subprogram name.

The ROUTINEN subprogram is supplied in the Natural system library SYSDB2. This subprogram is used to access the DB2 catalog to determine the formats of the user parameters defined for the Natural stored procedure or UDF. After the formats have been determined, they are stored in the Natural buffer pool. During subsequent invocations of the Natural stored procedure, the formats are then retrieved from the Natural buffer pool. This requires that at least READS SQL DATA is specified for Natural stored procedures or UDFs with PARAMETER STYLE DB2SQL.

The ROUTINEN subprogram is generated statically. The DBRM of ROUTINEN is bound as package in the COLLECTION SAGNDBROUTINENPACK. Before starting to access the DB2 catalog, the subprogram will save the CURRENT PACKAGESET and set SAGNDBROUTINENPACK to CURRENT PACKAGESET. After processing, the ROUTINEN subprogram will restore the CURRENT PACKAGESET saved.

Invoking a Natural Stored Procedure

If the caller of the Natural stored procedure with PARAMETER STYLE DB2SQL is a Natural program, the caller must use the CALLDBPROC statement with the specification CALLMODE=NONE, which is the default.

Error Handling

If a Natural runtime error occurs during the execution of a Natural stored procedure or UDF with PARAMETER STYLE DB2SQL, SQLSTATE is set to 38N99 and the diagnostic string contains the text of the Natural error message.

If an error occurs in the NDB server stub during the execution of the Natural stored procedure or UDF with PARAMETER STYLE DB2SQL, the SQLSTATE is set to 38S99 and the diagnostic string contains the text of the error message.

If the application wants to raise an error condition during the execution of a Natural stored procedure or UDF, the SQLSTATE parameter must be set to a value other than ‡00000Ÿ. See the DB2 SQL Guide for specifications of user errors in the SQLSTATE parameter.

Additionally, a text describing the errors can be placed in the DIAGNOSE parameter.

If a Natural table UDF wants to signal to DB2 that it has found no row to return, ’02000’ must be returned in the SQLSTATE parameter.

Lifetime of Natural Session

For a Natural UDF that contains the attributes DISALLOW PARALLEL and FINAL CALL, the NDB server stub retains the Natural session allocated earlier. This Natural session will then be reused by all subsequent UDF invocations until Natural encounters the final call.

Example of DB2SQL - Natural Stored Procedure

In a Natural stored procedure, define the parameters as shown in the example program below:

Example of DB2SQL - Natural Stored Procedure

```
DEFINE DATA PARAMETER
01 P1 ...
01 P2 ...
...
...
01 PN ...
01 N1 (I2)
01 N2 (I2)
...
...
01 Nn (I2)
PARAMETER USING DB2SQL_P
[ PARAMETER USING DBINFO_P ]   /*  only if DBINFO is defined
LOCAL
...
...
END-DEFINE
```

Example of DB2SQL - Natural UDF

In a Natural UDF, define the parameters as shown in the example program below:

Example of DB2SQL - Natural UDF

```
DEFINE DATA PARAMETER
01 PI1 ...  /* first input parameter
01 PI2 ...
...
...
01 PIn ...  /*  last input parameter
01 RS1...  /* first result parameter
...
...
01 RSn ...  /* last result parameter
01 N_PI1 (I2)  /* first NULL indicator
01 N_PI2 (I2)
...
...
01 N_Pin (I2)
01 N_RS1 (I2)
...
...
01 N_RSn (I2) /* last NULL indicator
PARAMETER USING DB2SQL_P /* function, specific, sqlstate, diagnose
PARAMETER
01 SCRATCHPAD A(4+nnn)  /* only if SCRATCHPAD nnn is specified
  01 REDEFINES SCRATCHPAD
02 SCRATCHPAD_LENGTH (I4)
02 ...
01 CALL_TYPE (I4) /* --- only if FINAL CALL is specified or table UDF


PARAMETER USING DBINFO_P   /* ---- only if DBINFO is specified
LOCAL
...
...
END-DEFINE
```


Writing a Natural Stored Procedure

This section provides a general guideline of how to write a Natural Stored Procedure and what to consider when writing it.

To write a Natural stored procedure

1. Determine the format and attributes of the parameters that are passed between the caller and the stored procedure.
   Consider creating a Natural PDA (parameter data area).
   Stored procedures do not support data groups and redefinition within their parameters.
2. Determine the PARAMETER STYLE of the stored procedure:
   GENERAL, GENERAL WITH NULL or DB2SQL.
   - If you use GENERAL WITH NULL, append the parameters to the Natural stored procedure by defining a NULL indicator array that contains a NULL indicator (I2) for each other parameter.
   - If you use DB2SQL, append the parameters of the Natural stored procedure by defining NULL indicators (one for each parameter), include the PDA DB2SQL_P and the PDA DBINFO_P (only with DBINFO specified), if desired.
     See also the relevant DB2 literature by IBM.
3. Decide which and how many result sets the stored procedure will return to the caller.
4. Code your stored procedure as a Natural subprogram.
   - **Returning result sets**
     To return result sets, code a SELECT statement with the WITH RETURN option.
     To return the whole result set, code an ESCAPE BOTTOM immediately after the SELECT.
     To return part of the result set code, an IF *COUNTER = 1 ESCAPE TOP END-IF immediately following the SELECT statement. This ensures that you do not process the first empty row that is returned by the SELECT WITH RETURN statement. To stop row processing, execute an ESCAPE BOTTOM statement.
     If you do not leave the processing loop initiated by the SELECT WITH RETURN via ESCAPE BOTTOM, the result set created is closed and nothing is returned to the caller.
   - **Attention when accessing other databases**
     You can access other databases (for instance Adabas) within a Natural stored procedure. However, keep in mind that your access to other databases is synchronized neither with the updates done by the caller of the stored procedure, nor with the updates done against DB2 within the stored procedure.
   - **NDB handling of COMMIT and ROLLBACK statements**
     DB2 does not allow a stored procedure to issue COMMIT or ROLLBACK statements (the execution of those statements puts the caller into a must-rollback state). Therefore, the NDB runtime handles those statements as follows when they are issued from a stored procedure:
     COMMIT against DB2 will be skipped.
     This allows the stored procedure to commit Adabas updates without getting a must-rollback state from DB2.
     ROLLBACK against DB2 will be skipped if it is created by Natural itself.
     ROLLBACK against DB2 will be executed if it is user-programmed.
     Thus, after a Natural error, the caller receives the Natural error information and not the unqualified must-rollback state. Additionally, this function ensures that, if the user program backs out the transaction, every database transaction of the stored procedure is backed out.
5. **For DB2 for OS/390 Version 5 and below:**
   Use the Procedure Maintenance function (see the relevant section) to define a stored procedure in the SYSIBM.SYSPROCEDURES table. To define a Natural stored procedure, enter the following data:

Column	Data
PROCEDURE	The name of your Natural subprogram representing your stored procedure.
LOADMOD	The name of your Natural for DB2 server stub module.
LINKAGE	Either blank or N as you decided at 2.
IBMREQD	N
PGM_TYPE	Depending on the specification of the MAIN parameter of the Natural for DB2 server stub module.
RESULT_SETS	As you decided at 3.
LANGUAGE	ASSEMBLER
PARMLIST	The description of the parameters you determined at 1. The first field of the PARMLIST **must** be defined as Natural internal parameter description STCB. It must be specified as a VARCHAR field with DATA INOUT and the following size: **274 + 13*N** where **N** is the number of parameters passed to the stored procedure (STCB not counted). If you have created a Natural PDA (parameter data area) for the stored procedure, you can easily derive the PARMLIST from your PDA by pressing PF5 on your PARMLIST screen of SYSDB2 and entering the library and the name of the PDA. In this case, the VARCHAR field for the parameter description is generated automatically.

For DB2 UDB for OS/390 Version 6 and above: Issue a CREATE PROCEDURE statement that defines your stored procedure, for example:

```
CREATE PROCEDURE <PROCEDURE>
  (INOUT  STCB       VARCHAR(274+13*N),
   INOUT  <PARM1>    <FORMAT>,
   INOUT  <PARM2>    <FORMAT>,
   INOUT  <PARM3>    <FORMAT>
   .
  )
  DYNAMIC RESULT SET <RESULT_SETS>
  EXTERNAL NAME <LOADMOD>
  LANGUAGE ASSEMBLE
  PROGRAM TYPE <PGM_TYPE>
  PARAMETER STYLE GENERAL <WITH NULLS depending on LINKAGE>;
```

The data specified in angle brackets (< >) correspond to the data listed in the table above, PARM1 - PARM3 and FORMAT depend on the call parameter list of the stored procedure.
See also Example Stored Procedure NDBPURGN, Member CR6PURGN.

6. Code your Natural program invoking the stored procedure via the CALLDBPROC statement.

   Code the parameters in the CALLDBPROC statement in the same sequence as they are specified in the stored procedure. Define the parameters in the calling program in a format that is compatible with the format defined in the stored procedure.

   If you use result sets, specify a RESULT SETS clause in the CALLDBPROC statement followed by a number of result set locator variables of FORMAT (I4). The number of result set locator variables should be the same as the number or result sets created by the stored procedure. If you specify fewer than are created, some result sets are lost. If you specify more than are created, the remaining result set locator variables are lost. The sequence of locator variables corresponds to the sequence in which the result sets are created by

the stored procedure.
Keep in mind that the fields into which the result set rows are read have to correspond to the fields used in the SELECT WITH RETURN statement that created the result set.

NDB - Writing a Natural UDF

This section provides a general guideline of how to write a Natural UDF and what to consider when writing it.

See also the section Writing a Natural Stored Procedure.

To write a Natural UDF

1. Determine the format and attributes of the parameters, which are passed between the caller and the stored procedure.
2. Create a Natural parameter data area (PDA).
3. Append the parameter definitions of the Natural UDF by defining NULL indicators (one for each parameter) and include the PDA DB2SQL_P.
4. If required, code a SCRATCHPAD area in the parameter list.
5. If required, code a call-type parameter.
   If you have specified DBINFO, include the PDA DBINFO_P. See also the relevant DB2 literature by IBM.
6. Code your UDF as a Natural subprogram and consider the following:
   - **Attention when accessing other databases**
     You can access other databases (for example, Adabas) within a Natural UDF. However, keep in mind that your access to other databases is synchronized neither with the updates done by the caller of the stored procedure, nor with the updates done against DB2 within the stored procedure.
   - **NDB handling of COMMIT and ROLLBACK statements**
     DB2 does not allow a stored procedure to issue COMMIT or ROLLBACK statements; the execution of these statements results in a must-rollback state. If a Natural stored procedure issues a COMMIT or ROLLBACK, the NDB runtime processes these statements as follows:
     COMMIT against DB2 is skipped.
     This allows the stored procedure to commit Adabas updates without entering a must-rollback state by DB2.
     ROLLBACK against DB2 is skipped if it is implicitly issued by the Natural runtime.
     ROLLBACK against DB2 is executed if it is user-programmed.
     Thus, after a Natural error, the caller receives a corresponding Natural error message text, but does not enter an unqualified must-rollback state. Additionally, this reaction ensures that every database transaction the stored procedure performs is backed out if the user program backs out the transaction.
7. Issue a CREATE FUNCTION statement that defines your UDF, for example:

```
CREATE FUNCTION <FUNCTION>
  ([PARM1]    <FORMAT>,
   [PARM2]    <FORMAT>,
   [PARM3]    <FORMAT>

  )
  RETURNS <FORMAT>

  EXTERNAL NAME <LOADMOD>
  LANGUAGE ASSEMBLE
  PROGRAM TYPE <PGM TYPE>
  PARAMETER STYLE DB2SQL
.
.
.;
```

In the example above, the variable data are enclosed in angle brackets (< >) and refer to the keywords preceding the brackets. Specify a valid value, for example:

LOADMOD denotes the NDB server stub module, for example, NDB41SRV. PARM1 - PARM3 and FORMAT relate to the call parameter list of the UDF. See also the Example UDF.

8. Code a Natural program containing SQL statements that invoke the UDF.

   Specify the parameters of the Natural UDF invocation in the same sequence as specified in the Natural UDF definition. The format of the parameters in the calling program must be compatible with the format defined in the Natural UDF.

NDB - Example Stored Procedure

This section describes the example stored procedure NDBPURGN, a Natural subprogram which purges Natural objects from the buffer pool used by the Natural stored procedures server.

This section covers the following topics:

- Members of NDBPURGN
- Defining the Stored Procedure
- Stored Procedure Control Block (STCB)

Members of NDBPURGN

The example stored procedure NDBPURGN comprises the following text members which are stored in the Natural system library SYSDB2:

Member	Explanation
CR5PURGN	Input member for SYSDB2 ISQL. Contains SQL statements used to declare NDBPURGN in DB2 Version 5.
CR6PURGN	Input member for SYSDB2 ISQL. Contains SQL statements used to declare NDBPURGN in DB2 Version 6 and above.
NDBPURGP	The client (Natural) program which • Requests the name of the program to be purged and the library where it resides, • Invokes the stored procedure NDBPURGN and • Reports the outcome of the request.
NDBPURGN	The stored procedure which purges objects from the buffer pool. NDBPURGN invokes the user exit USR0340N supplied in the Natural system library SYSEXT. Therefore, USR0340N must be available in the library defined as the steplib for the execution environment.

Defining the Stored Procedure NDBPURGN

To define the example stored procedure NDBPURGN

- Define the stored procedure in the DB2 catalog by using the SQL statements provided as text members CR5PURGN (for DB2 Version 5) and CR6PURGN (for DB2 Version 6).
- Specify the name of the Natural stored procedure stub (here: NDB41SRV) as LOADMOD (V5) or EXTERNAL NAME (V6). The Natural stored procedure stub is generated during the installation by assembling the NDBSTUB macro.
- As the first parameter, pass the internal Natural parameter STCB to the stored procedure.
  The STCB parameter is a VARCHAR field which contains information required to invoke the stored procedure in Natural:
  - The program name of the stored procedure and the library where it resides,
  - The description of the parameters passed to the stored procedure and
  - The error message created by Natural if the stored procedure fails during the execution.

The STCB parameter is generated automatically by the CALLMODE=NATURAL clause of the CALLDBPROC statement and is removed from the parameters passed to the Natural stored procedure by the server stub. Thus, STCB is invisible to the caller and the stored procedure. However, if a non-Natural client intends to call a Natural stored procedure, the client has to pass the STCB parameter explicitly. See also Stored Procedure Control Block below.

Stored Procedure Control Block (STCB)

Below is the Stored Procedure Control Block (STBC) generated by the CALLMODE=NATURAL clause as generated by the stored procedure NDBPURGN before and after execution. Changed values are emphasized in boldface:

STCB before Execution:

```
004C82   0132F0F3  F0F6E2E3  C3C2F3F1  F040C8C7  *..0306STCB310 HG*    11097D42
004C92   D2404040  4040C8C7  D2404040  4040D5C4  *K     SAG     ND*    11097D52
004CA2   C2D7E4D9  C7D74040  40404040  4040F0F5  *BPURGP        05*    11097D62
004CB2   F7F0D5C4  C2D7E4D9  C7D5F0F0  F0F6F0F9  *70NDBPURGN000609*    11097D72
004CC2   F9F9F940  40404040  40404040  40404040  *999              *    11097D82
004CD2   40404040  40404040  40404040  40404040  *                 *    11097D92
004CE2   40404040  40404040  40404040  40404040  *                 *    11097DA2
004CF2   40404040  40404040  40404040  40404040  *                 *    11097DB2
004D02   40404040  40404040  40404040  40404040  *                 *    11097DC2
004D12   40404040  40404040  40404040  40404040  *                 *    11097DD2
004D22   40404040  40404040  40404040  40404040  *                 *    11097DE2
004D32   40404040  40404040  40404040  40404040  *                 *    11097DF2
004D42   40404040  40404040  40404040  40404040  *                 *    11097E02
004D52   40404040  40404040  40404040  40404040  *                 *    11097E12
004D62   40404040  40404040  40404040  40404040  *                 *    11097E22
004D72   40404040  40404040  40404040  40404040  *                 *    11097E32
004D82   40404040  40404040  40404040  40404040  *                 *    11097E42
004D92   40404040  D4C1F86B  F0D4C1F4  F06BF0D4  *    MA8,0MA40,0M*    11097E52
004DA2   C2F26BF0  D4C2F26B  F0D4C9F2  6BF0D4C9  *I2,0MI2,0MI2,0MI*    11097E62
004DB2   F26BF04B                               *2,0.             *    11097E72
```

STCB after Execution:

```
004C82   0132F0F3  F0F6E2E3  C3C2F3F1  F040C8C7  *..0306STCB310 HG*    11097D42
004C92   D2404040  4040C8C7  D2404040  4040D5C4  *K     SAG     ND*    11097D52
004CA2   C2D7E4D9  C7D74040  40404040  4040F0F5  *BPURGP        05*    11097D62
004CB2   F7F0D5C4  C2D7E4D9  C7D5F0F0  F0F6F0F0  *70NDBPURGN000600*    11097D72
004CC2   F0F0F040  40404040  40404040  40404040  *000              *    11097D82
004CD2   40404040  40404040  40404040  40404040  *                 *    11097D92
004CE2   40404040  40404040  40404040  40404040  *                 *    11097DA2
004CF2   40404040  40404040  40404040  40404040  *                 *    11097DB2
004D02   40404040  40404040  40404040  40404040  *                 *    11097DC2
004D12   40404040  40404040  40404040  40404040  *                 *    11097DD2
004D22   40404040  40404040  40404040  40404040  *                 *    11097DE2
004D32   40404040  40404040  40404040  40404040  *                 *    11097DF2
004D42   40404040  40404040  40404040  40404040  *                 *    11097E02
004D52   40404040  40404040  40404040  40404040  *                 *    11097E12
004D62   40404040  40404040  40404040  40404040  *                 *    11097E22
004D72   40404040  40404040  40404040  40404040  *                 *    11097E32
004D82   40404040  40404040  40404040  40404040  *                 *    11097E42
004D92   40404040  D4C1F86B  F0D4C1F4  F06BF0D4  *    MA8,0MA40,0M*    11097E52
004DA2   C2F26BF0  D4C2F26B  F0D4C9F2  6BF0D4C9  *I2,0MI2,0MI2,0MI*    11097E62
004DB2   F26BF04B                               *2,0.             *    11097E72
```

NDB - Example Natural UDF

This section describes the example UDF NAT.DEM2UDFN, a Natural subprogram used to calculate the product of two numbers.

The example UDF NAT.DEM2UDF comprises the following members that are supplied in the Natural system library SYSDB2:

Member	Explanation
DEM2CUDF	Contains SQL statements used to create DEM2UDFN (see below).
DEM2UDFP	The client (Natural) program that • Fetches rows from the NAT.DEMO table, • Invokes the UDF NAT.DEM2UDFN (see below) in the WHERE clause, and • Displays the rows fetched.
DEM2UDFN	The UDF that builds the product of two numbers. DEM2UDFN has to be copied to the Natural library NAT on the FUSER in the executing environment.

NDB - Security

DB2 provides an authorization ID for the execution of a Natural stored procedure or Natural UDF to control access to non-SQL resources with an external security product, such as RACF. The NDB server stub uses this authorization ID to perform an implicit LOGON within the Natural session created for the Natural stored procedure or the Natural UDF. So, if the Natural stored procedure or Natural UDF is to be executed in a Natural Security environment, ensure that the authorization ID used has been defined in the FSEC file.

As shown in the table below, the authorization DB2 provides depend on the definition of the SECURITY attribute specified for the Natural stored procedure or UDF:

SECURITY Attribute	DB2 Authorization ID of:
DB2	The address space in which the Natural stored procedure or Natural UDF is executed.
USER	The process (caller) that invokes the Natural stored procedure or Natural UDF
DEFINER	The owner of the Natural stored procedure or UDF.

NDB - Interface Subprograms

Several Natural and non-Natural subprograms are available to provide you with either internal information from the Natural interface to DB2 or specific functions that are not available within the interface itself.

From within a Natural program, Natural subprograms are invoked with the CALLNAT statement and non-Natural subprograms are invoked with the CALL statement.

This section covers the following topics:

- Natural Subprograms
- DB2SERV Interface

Natural Subprograms

The following Natural subprograms are provided:

- NDBDBRM Subprogram
- NDBDBR2 Subprogram
- NDBERR Subprogram
- NDBISQL Subprogram
- NDBNOERR Subprogram
- NDBNROW Subprogram
- NDBSTMP Subprogram

Subprogram	Function
NDBDBRM	Checks whether a Natural program contains SQL access and whether it has been modified for static execution.
NDBDBR2	Checks whether a Natural program contains SQL access and whether it has been modified for static execution.
NDBERR	Provides diagnostic information on the most recently executed SQL call.
NDBISQL	Executes SQL statements in dynamic mode.
NDBNOERR	Suppresses normal Natural error handling.
NDBNROW	Obtains the number of rows affected by a Natural SQL statement.
NDBSTMP	Provides a DB2 TIMESTAMP column as an alphanumeric field and vice versa.

All these subprograms are provided in the Natural system library SYSDB2 . Copy them to the SYSTEM or steplib library, or to any library where they are needed. In addition, the subprogram DBTLIB2N and the subroutine DBDL219S have to be copied from SYSDB2. They are used by NDBDBRM and NDBDBR2. The corresponding parameters must be defined in a DEFINE DATA statement.

NDBDBRM Subprogram

The Natural subprogram NDBDBRM is used to check whether a Natural program contains SQL access and whether it has been modified for static execution. It is also used to obtain the corresponding DBRM (database request module) name from the header of a Natural program generated as static (see also Preparing Natural Programs for Static Execution).

A sample program called CALLDBRM is provided on the installation tape; it demonstrates how to invoke NDBDBRM. A description of the call format and of the parameters is provided in the text member NDBDBRMT.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBDBRM' #LIB #MEM #DBRM #RESP
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
#LIB	A8	Contains the name of the library of the program to be checked.
#MEM	A8	Contains the name of the program (member) to be checked
#DBRM	A8	Returns the DBRM name.
#RESP	I2	Returns a response code. The possible codes are listed below.

The #RESP parameter can contain the following response codes:

Code	Explanation
0	The member #MEM in library #LIB has SQL access; it is static if #DBRM contains a value.
-1	The member #MEM in library #LIB has no SQL access.
-2	The member #MEM in library #LIB does not exist.
-3	No library name has been specified.
-4	No member name has been specified.
-5	The library name must start with a letter.
<-5	Further negative response codes correspond to error numbers of Natural error messages.
> 0	Positive response codes correspond to error numbers of Natural Security messages.

NDBDBR2 Subprogram

The Natural subprogram NDBDBR2 is used to check whether a Natural program contains SQL access and whether it has been modified for static execution. It is also used to obtain the corresponding DBRM name from the header of a Natural program generated as static (see also Preparing Natural Programs for Static Execution) and the time stamp generated by the precompiler.

A sample program called CALLDBR2 is provided on the installation tape; it demonstrates how to invoke NDBDBR2. A description of the call format and of the parameters is provided in the text member NDBDBR2T.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBDBR2' #LIB #MEM #DBR2 #TIMESTAMP #PCUSER #PCRELLEV #ISOLLEVL #DATEFORM #TIMEFORM #RESP
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
#LIB	A8	Contains the name of the library of the program to be checked.
#MEM	A8	Contains the name of the program (member) to be checked
#DBR2	A8	Returns the DBR2 name.
#TIMESTAMP	B8	Consistency token generated by precompiler
#PCUSER	A1	User ID used at precomplile (only SQL/DS)
#PCRELLEV	A1	Release level of precompiler (only SQL/DS)
#ISOLLEVL	A1	Precomplier isolation level (only SQL/DS)
#DATEFORM	A1	Date format (only SQL/DS)
#TIMEFORM	A1	Time format (only SQL/DS)
#RESP	I2	Returns a response code. The possible codes are listed below.

The #RESP parameter can contain the following response codes:

Code	Explanation
0	The member #MEM in library #LIB has SQL access; it is static if #DBRM contains a value.
-1	The member #MEM in library #LIB has no SQL access.
-2	The member #MEM in library #LIB does not exist.
-3	No library name has been specified.
-4	No member name has been specified.
-5	The library name must start with a letter.
<-5	Further negative response codes correspond to error numbers of Natural error messages.
> 0	Positive response codes correspond to error numbers of Natural Security messages.

NDBERR Subprogram

The Natural subprogram NDBERR replaces Function **E** of the DB2SERV interface, which is still provided but no longer documented. It provides diagnostic information on the most recent SQL call. It also returns the database type which returned the error. NDBERR is typically called if a database call returns a non-zero SQL code (which means a NAT3700 error); see also Error Handling in Statements and System Variables.

A sample program called CALLERR is provided on the installation tape; it demonstrates how to invoke NDBERR. A description of the call format and of the parameters is provided in the text member NDBERRT.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBERR' #SQLCODE #SQLSTATE #SQLCA #DBTYPE
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
SQLCODE	I4	Returns the SQL return code.
SQLSTATE	A5	Returns a return code for the output of the most recently executed SQL statement.
SQLCA	A136	Returns the SQL communication area of the most recent DB2 access.
DBTYPE	B1	Returns the identifier (in hexadecimal format) for the currently used database (where X'02' identifies DB2).

NDBISQL Subprogram

The Natural subprogram NDBISQL is used to execute SQL statements in dynamic mode. The SELECT statement and all SQL statements which can be prepared dynamically (according to the DB2 literature by IBM) can be passed to NDBISQL.

A sample program called CALLISQL is provided on the installation tape; it demonstrates how to invoke NDBISQL. A description of the call format and of the parameters is provided in the text member NDBISQLT.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBISQL'#FUNCTION #TEXT-LEN #TEXT (*) #SQLCA #RESPONSE
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
#FUNCTION	A8	For valid functions, see below.
#TEXT-LEN	I2	Length of the SQL statement or of the buffer for the return area.
#TEXT	A1(1:V)	Contains the SQL statement or receives the return code.
#SQLCA	A136	Contains the SQLCA.
#RESPONSE	I4	Returns a response code.

Valid functions for the #FUNCTION parameter are:

Function	Parameter	Explanation
CLOSE		Closes the cursor for the SELECT statement.
EXECUTE	#TEXT-LEN #TEXT (*)	Executes the SQL statement. Contains the length of the statement. Contains the SQL statement. The first two characters must be blank.
FETCH	#TEXT-LEN #TEXT (*)	Returns a record from the SELECT statement. Size of #TEXT (in bytes). Buffer for the record.
TITLE	#TEXT-LEN #TEXT (*)	Returns the header for the SELECT statement. Size of #TEXT (in bytes); receives the length of the header (= length of the record). Buffer for the header line.

The #RESPONSE parameter can contain the following response codes:

Code	Function	Explanation
5	EXECUTE	The statement is a SELECT statement.
6	TITLE, FETCH	Data are truncated; only set on first TITLE or FETCH call.
100	FETCH	No record / end of data.
-2		Unsupported data type (for example, GRAPHIC).
-3	TITLE, FETCH	No cursor open; probably invalid call sequence or statement other than SELECT.
-4		Too many columns in result table.
-5		SQL code from call.
-6		Version mismatch.
-7		Invalid function.
-10		Interface not available.
-11	EXECUTE	First two bytes of statement not blank.

Call Sequence

The first call must be an EXECUTE call. If the statement is a SELECT statement (that is, response code 5 is returned), any sequence of TITLE and FETCH calls can be used to retrieve the data. A response code of 100 indicates the end of the data.

The cursor must be closed with a CLOSE call.

Function code EXECUTE implicitly closes a cursor which has been opened by a previous EXECUTE call for a SELECT statement.

In TP environments, no terminal I/O can be performed between an EXECUTE call and any TITLE, FETCH or CLOSE call that refers to the same statement.

NDBNOERR Subprogram

The Natural subprogram NDBNOERR is used to suppress Natural NAT3700 errors caused by the next SQL call. This allows a program controlled continuation if an SQL statement produces a non-zero SQL code. After the SQL call has been performed, NDBERR is used to investigate the SQL code; see also Error Handling in Statements and System Variables.

A sample program called CALLNOER is provided on the installation tape; it demonstrates how to invoke NDBNOERR. A description of the call format and of the parameters is provided in the text member NDBNOERT.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBNOERR'
```

There are no parameters provided with this subprogram.

Note:
Only NAT3700 errors (that is, non-zero SQL response codes) are suppressed, and also only errors caused by the next following SQL call.

Restrictions with Database Loops

- If NDBNOERR is called before a statement that initiates a database loop and an initialization error occurs, no processing loop will be initiated, unless the IF NO RECORDS FOUND clause has been specified.
- If NDBNOERR is called within a database loop, it does not apply to the processing loop itself, but only to the SQL statement subsequently executed inside this loop.

NDBNROW Subprogram

The Natural subprogram NDBNROW is used to obtain the number of rows affected by the Natural SQL statements Searched UPDATE, Searched DELETE, and INSERT. The number of rows affected is read from the SQL communication area (SQLCA). A positive value represents the number of affected rows, whereas a value of minus one (-1) indicates that all rows of a table in a segmented tablespace have been deleted (see also the Natural system variable *NUMBER as described in Natural System Variables).

A sample program called CALLNROW is provided on the installation tape; it demonstrates how to invoke NDBNROW. A description of the call format and of the parameters is provided in the text member NDBNROWT.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBNROW' #NUMBER
```

The parameter #NUMBER (I4) contains the number of affected rows.

NDBSTMP Subprogram

For DB2, Natural provides a TIMESTAMP column as an alphanumeric field (A26) of the format *YYYY-MM-DD-HH.MM.SS.MMMMMM*.

Since Natural does not yet support computation with such fields, the Natural subprogram NDBSTMP is provided to enable this kind of functionality. It converts Natural time variables to DB2 time stamps and vice versa and performs DB2 time stamp arithmetics.

A sample program called CALLSTMP is provided on the installation tape; it demonstrates how to invoke NDBSTMP. A description of the call format and of the parameters is provided in the text member NDBSTMPT.

The functions available are:

Code	Explanation
ADD	Adds time units (labeled durations) to a given DB2 time stamp and returns a Natural time variable and a new DB2 time stamp.
CNT2	Converts a Natural time variable (format T) into a DB2 time stamp (column type TIMESTAMP) and labeled durations.
C2TN	Converts a DB2 time stamp (column type TIMESTAMP) into a Natural time variable (format T) and labeled durations.
DIFF	Builds the difference between two given DB2 time stamps and returns labeled durations.
GEN	Generates a DB2 time stamp from the current date and time values of the Natural system variable *TIMX and returns a new DB2 time stamp.
SUB	Subtracts labeled durations from a given DB2 time stamp and returns a Natural time variable and a new DB2 time stamp.
TEST	Tests a given DB2 time stamp for valid format and returns TRUE or FALSE.

Note:
Labeled durations are units of year, month, day, hour, minute, second and microsecond.

DB2SERV Interface

DB2SERV is an Assembler program entry point which can be called from within a Natural program.

DB2SERV performs either of the following functions:

- Function D, which performs the SQL statement EXECUTE IMMEDIATE;
- Function P, which is used to establish the DB2 connection (TSO and batch mode only).

The parameter or variable values returned by each of these functions are checked for their format, length, and number.

Function D

Function **D** performs the SQL statement EXECUTE IMMEDIATE. This allows SQL statements to be issued from within a Natural program.

The SQL statement string that follows the EXECUTE IMMEDIATE statement must be assigned to the Natural program variable STMT. It must contain valid SQL statements allowed with the EXECUTE IMMEDIATE statement as described in the relevant IBM literature. Examples can be found below and in the demonstration programs DEM2* in the Natural system library SYSDB2.

Note:
The conditions that apply to issuing the Natural END TRANSACTION or BACKOUT TRANSACTION statements also apply when issuing the SQL COMMIT or ROLLBACK statements: see the relevant sections in Statements and System Variables (Natural SQL Statements).

Command Syntax

```
CALL 'DB2SERV' 'D' STMT STMTL SQLCA RETCODE
```

The variables used in this command are described in the following table:

Variable	Format/Length	Explanation
STMT	A*nnn*	Contains a command string which consists of SQL syntax as described above.
STMTL	I2	Contains the length of the string defined in STMT.
SQLCA	A136	Returns the current contents of the SQL communication area.
RETCODE	I2	Returns an interface return code. The following codes are possible: 0 No warning or error occurred. 4 SQL statement produced an SQL warning. 8 SQL statement produced an SQL error. 12 Internal error occurred; the corresponding Natural error message number can be displayed with SQLERR.

The current contents of the SQLCA and an interface return code (RETCODE) are returned. The SQLCA is a collection of variables that are used by DB2 to provide an application program with information on the execution of its SQL statements.

The following two examples show you how to use DB2SERV with Function **D**:

Example of Function D - DEM2CREA:

```
************************************************************************
*  DEM2CREA - CREATE TABLE NAT.DEMO                                    *
************************************************************************
*
DEFINE DATA
LOCAL USING DEMSQLCA
LOCAL
*                                    Parameters for DB2SERV
1 STMT         (A250)
1 STMTL        (I2)     CONST <250>
1 RETCODE      (I2)
*
END-DEFINE
*
COMPRESS  'CREATE TABLE NAT.DEMO'
  '(NAME        CHAR(20)     NOT NULL,'
  ' ADDRESS     VARCHAR(100) NOT NULL,'
  ' DATEOFBIRTH DATE         NOT NULL,'
  ' SALARY      DECIMAL(6,2),'
  ' REMARKS     VARCHAR(500))'
  INTO STMT
CALL 'DB2SERV' 'D' STMT STMTL SQLCA RETCODE
*
END TRANSACTION
*
IF RETCODE = 0
  WRITE 'Table NAT.DEMO created'
ELSE
  FETCH 'SQLERR'
END-IF
END
************************************************************************
```

Note:
The functionality of the DB2SERV Function **D** is also provided with the PROCESS SQL statement: see the relevant section in Statements and System Variables (Natural SQL Statements).

Example of Function D - DEM2SET:

```
************************************************************************
*  DEM2SET - Set Current SQLID                                         *
************************************************************************
*
DEFINE DATA
LOCAL USING DEMSQLCA
LOCAL
*                                    Parameter for DB2SERV
1 STMT         (A250)
1 STMTL        (I2)     CONST <250>
1 RETCODE      (I2)
1 OLDSQLID     (A8)
1 NEWSQLID     (A8)
*
END-DEFINE
*
SELECT DISTINCT CURRENT SQLID
  INTO OLDSQLID
  FROM SYSIBM.SYSTABLES
ESCAPE BOTTOM
END-SELECT
*
MOVE  'SET CURRENT SQLID="PROD"';
  TO   STMT
CALL 'DB2SERV' 'D' STMT STMTL SQLCA RETCODE
*
IF RETCODE > 0
  FETCH 'SQLERR'
ELSE
  SELECT DISTINCT CURRENT SQLID
    INTO NEWSQLID
    FROM SYSIBM.SYSTABLES
  ESCAPE BOTTOM
  END-SELECT
*
  WRITE ' Old SQLID was :' OLDSQLID
  WRITE ' New SQLID is  :' NEWSQLID
END-IF
*
END
************************************************************************
```

When using SET CURRENT SQLID, the creator name of a table can be substituted by the current SQLID. This enables you to access identical tables with the same table name but with different creator names. Thus, table names must not be qualified by a creator name if this is to be substituted by the SQLID.

In all supported TP-monitor environments, the SQLID can then be kept across terminal I/Os until either the end of the session or its resetting via DB2SERV.

Function P

Function **P** invokes an Assembler module named NDBPLAN, which is used to establish and/or terminate the DB2 connection under TSO and in batch mode. This allows a Natural application to perform plan switching under TSO and in batch mode.

The program DEM2PLAN is an example of the use of DB2SERV with Function **P**.

The name of the current DB2 subsystem (#SSM) and the name of the new application plan (#PLAN) must be specified. In addition, an interface return code (#RETCODE) and the DB2 reason code (#REASON) are returned.

Command Syntax

```
CALL 'DB2SERV' 'P' #SSM #PLAN #RETCODE #REASON
```

Variable	Format/Length	Explanation
#SSM	A4	Contains the name of the current DB2 subsystem.
#PLAN	A8	Contains the new plan name.
#RETCODE		Returns an interface return code. The following codes are possible: 0 No warning or error occurred. 12 The specified new application plan is not scheduled. 99 The current environment is not a CAF environment. *nnn* Return code from the CAF interface (see also the relevant DB2 literature by IBM).
#REASON	I4	Returns the reason code of the CAF interface (see also the relevant DB2 literature by IBM).

Example of Function P - DEM2PLAN:

```
************************************************************************
*  DEM2PLAN - Switch application plan under TSO/Batch with CAF interface *
************************************************************************
*
DEFINE DATA
LOCAL
*                                  Parameter for DB2SERV
01 #SSM         (A4))    CONST <'DB2'>
01 #PLAN        (A8
01 #RETCODE     (I2)
01 #REASON      (I4)
*
END-DEFINE
*
INPUT 'PLEASE ENTER NEW PLAN NAME' #PLAN (AD='_'I)
*
END TRANSACTION
*
CALL 'DB2SERV' 'P' #SSM #PLAN #RETCODE #REASON
*
DECIDE FOR FIRST VALUE OF #RETCODE
*
  VALUE  0
    IGNORE
  VALUE  99
    INPUT 12/23 'This is not a CAF environment !!'
  VALUE  8,12
    INPUT 12/18 'New plan not scheduled, reason code'
                #REASON (AD=OI EM=H(4))
  NONE
    INPUT 12/15 'CAF interface error'
                #RETCODE (AD=OI EM=Z(3))
                'with reason code'
                #REASON (AD=OI EM=H(4))
*
END-DECIDE
*
END
************************************************************************
```

Important:
Plan switching under TSO and in batch mode is possible with the CAF interface only; see also the section Plan Switching under TSO and in Batch Mode.

Natural File Server for DB2

In all supported TP-monitor environments (CICS, IMS/TM, and TSO), the Natural interface to DB2 provides an intermediate work file, referred to as the File Server, to prevent database selection results from being lost with each terminal I/O. Exception: Com-plete.

This section covers the following topics:

- Concept of the File Server
- Installing the File Server
- Logical Structure of the File Server

Concept of the File Server

To avoid reissuing the selection statement used and repositioning the cursors, Natural writes the results of a database selection to an intermediate file. The saved selected rows, which may be required later, are then managed by Natural as if the facilities for conversational processing were available. This is achieved by automatically scrolling the intermediate file for subsequent screens, maintaining position in the work file rather than in DB2.

All rows of all open cursors are rolled out to the file server before the first terminal I/O operation. Subsequently, all data is retrieved from this file if Natural refers to one of the cursors which were previously rolled out (see the description of roll out in Logical Structure of File Server below).

If a row is to be updated or deleted, the row is first checked to see if it has been updated in the meantime by some other process. This is done by reselecting and fetching the row from the DB2 database, and then comparing it with the original version as retrieved from the file server. If the row is still unchanged, the update or delete operation can be executed. If not, a corresponding error message is returned. The reselection required when updating or deleting a row is possible in both dynamic mode and static mode.

Only the fields which are stored in the file server are checked for consistency against the record retrieved from DB2.

As the row must be uniquely identified, the Natural view must contain a field for which a unique row has been created. This field must be defined as a unique key in DB2. In a Natural DDM, it will then be indicated as a unique key via the corresponding Natural-specific short name.

Installing the File Server

The size of a row which can be written to the file server is limited to 32 KB or 32767 bytes. If a row is larger, a corresponding error message is returned.

The File Server can use either a VSAM RRDS file or the Software AG Editor buffer pool as storage medium to save selected rows of DB2 tables.

This section covers the following topics:

- Installing the File Server using VSAM
- Installing the File Server using the Editor Buffer Pool

Installing the File Server - VSAM

The file server is installed via a batch job, which defines and formats the intermediate file. Samples of this batch job are supplied on the installation tape as described in the relevant section.

Defining the Size of the File Server

The file server is created by defining an RRDS VSAM file using AMS (Access Method Services). Its physical size and its name must be specified.

Formatting the File Server

The file server is formatted by a batch job, which requires five input parameters specified by the user, and which formats the file server according to these parameters. The parameters specify:

1. The number of blocks to be formatted (logical size of the VSAM file); this value is taken from the first parameter of the RECORD subcommand of the AMS DEFINE CLUSTER command.
2. The number of users that can log on to Natural concurrently.
3. The number of formatted blocks to be defined as primary allocation per user.
4. The number of formatted blocks to be used as secondary allocation per user.
5. The maximum number of file server blocks to be allocated by each user. If this number is exceeded, a corresponding Natural error message is returned.

Immediately before the first access to the file server, a file server directory entry is allocated to the Natural session and the amount of blocks specified as primary allocation is allocated to the Natural session.

The primary allocation is used as intermediate storage for the result of a database selection and should be large enough to accommodate all rows of an ordinary database selection. Should more space in the file server be required for a large database selection, the file server modules allocate a secondary allocation equal to the amount that was specified for secondary allocation when the file server was formatted.

Thus, a secondary area is allocated only when your current primary allocation is not large enough to contain all of the data which must be written to the intermediate file. The number of secondary allocations allowed depends upon the maximum number of blocks you are allowed to allocate. This parameter is also specified when formatting the file server.

The number of blocks defined as the secondary allocation is allocated repeatedly, until either all selected data has been written to the file or the maximum number of blocks you are allowed to allocate is exceeded. If so, a corresponding Natural error message is returned. When the blocks received as a secondary allocation are no longer needed (that is, once the Natural loop associated with this allocation is closed), they are returned to the free blocks pool of the file server.

Your primary allocation of blocks, however, is always allocated to you, until the end of your Natural session.

Changes Required for a Multi-Volume File Server

To minimize channel contention or bottlenecks that can be caused by placing a large and heavily used file server on a single DASD volume, you can create a file server that spans several DASD volumes.

To create and format such a file server, two changes are needed in the job that is used to define the VSAM cluster:

1. Change VOLUME () to VOLUMES (vol1,vol2,...).
2. Divide the total number of records required for the file (as specified with the first format job parameter) by the number of volumes specified above. The result of the calculation is used for the RECORDS parameter of the DEFINE CLUSTER command.

This means that in the file server format job, the value of the first parameter is the result of multiplying two parameters taken from the DEFINE CLUSTER command: RECORDS and VOLUMES.

Installing the File Server - Editor Buffer Pool

The Software AG Editor buffer pool is used as storage medium when EBPFSRV=ON is set in the NDBPARM module. In this case, the primary, secondary and maximum allocation amounts for the file server are specified by EBPPRAL, EBPSEC, EBPMAX parameters of the NDBPRM macro. Before NDB tries to write data from a Natural user session to the file server for the first time, a Software AG Editor buffer pool logical file is allocated with the Natural terminal identifier as user name and the number 2240 as session number.

The operation of the file server is in this case depending on the definition of the Software AG Editor buffer pool as described in the Natural Operations for Mainframes documentation.

The number of logical files for the buffer pool limits the number of users concurrently accessing the file server. The number of work file blocks limits the amount of data to be saved at a specific moment. (You also have to consider that there are other users than NDB of the Software AG Editor.)

However, using the Software AG Editor buffer pool as storage medium for the file server enables NDB to run in a Sysplex environment.

If you like to use the file server in a sysplex environment, it is recommended to use the Software AG Editor buffer pool as storage medium.

Logical Structure of the File Server

Immediately before a Natural user session accesses the file server, a file server directory entry (VSAM) or a logical file (Software AG Editor buffer pool) is allocated to the Natural user session and the number of blocks specified as primary allocation is reserved until the end of the session.

Generally, the file server is only used when a terminal I/O occurs within an active READ, FIND, or SELECT loop, where database selection results would be lost. Before each terminal I/O operation, Natural checks for any open cursors. For each non-scrollable cursor found, all remaining rows are retrieved from DB2 and written to an intermediate file. For each scrollable cursor, all rows are retrieved from DB2 and written to an intermediate file. In the NDB documentation, this process is referred to as cursor roll out.

For each cursor roll out (scrollable and non-scrollable), a logical file is opened to hold all the rows fetched from this cursor. The space for the intermediate file is managed within the space allocated to your session. The logical file is then positioned on the row that was CURRENT OF CURSOR when the terminal I/O occurred.

Subsequent requests for data are then satisfied by reading the rows directly from the intermediate file. The database is no longer involved, and DB2 is only used for update, delete or store operations.

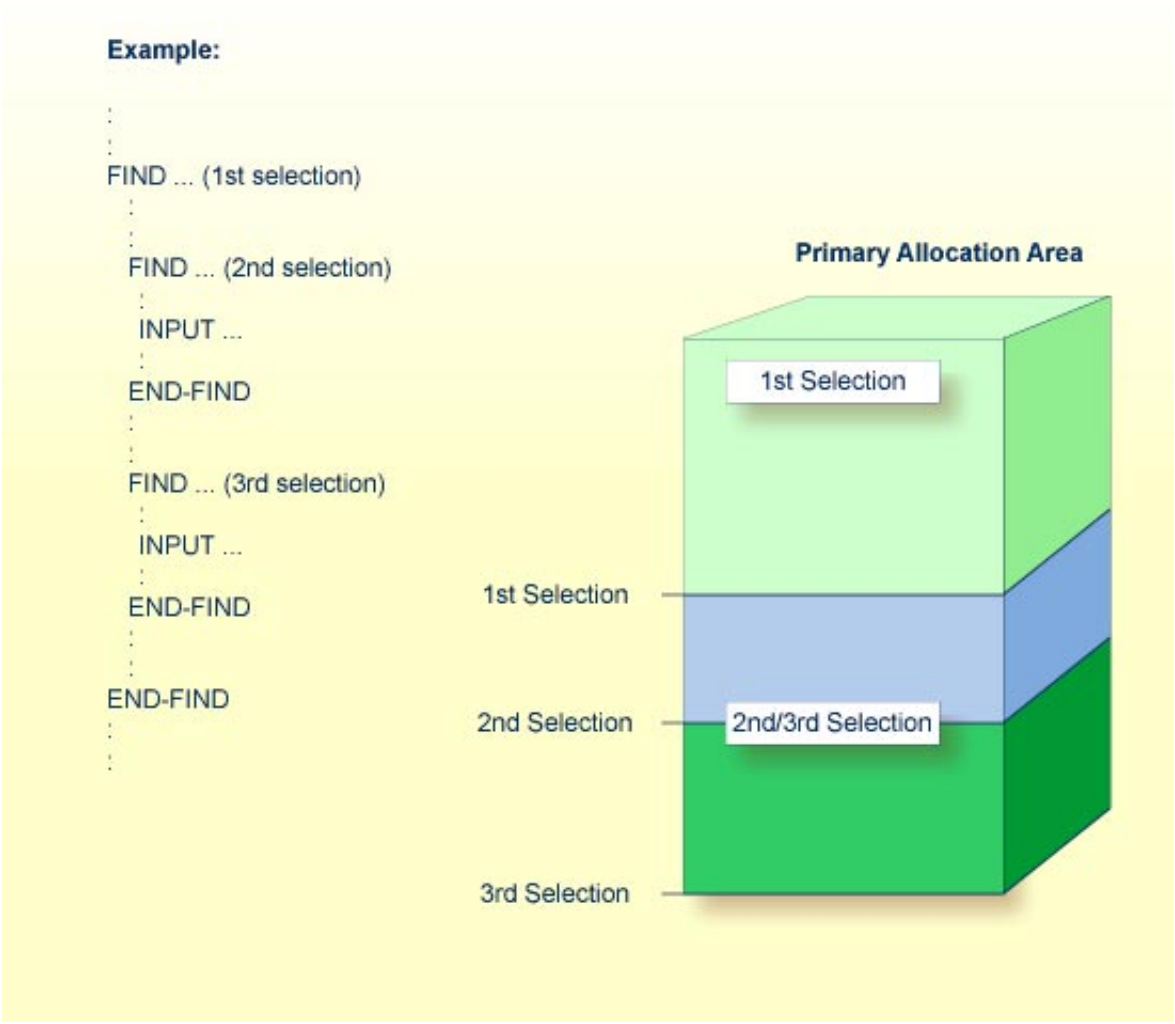
Positioned UPDATE and/or Positioned DELETE statements against rolled-out scrollable cursors are performed against the DB2 base table and against the logical file on the file server.

Once the corresponding processing loop in the application has been closed, the file is no longer needed and the blocks it occupies are returned to your pool of free blocks. From here, the blocks are returned to the free blocks pool of the file server, so that you are left with your primary allocation only.

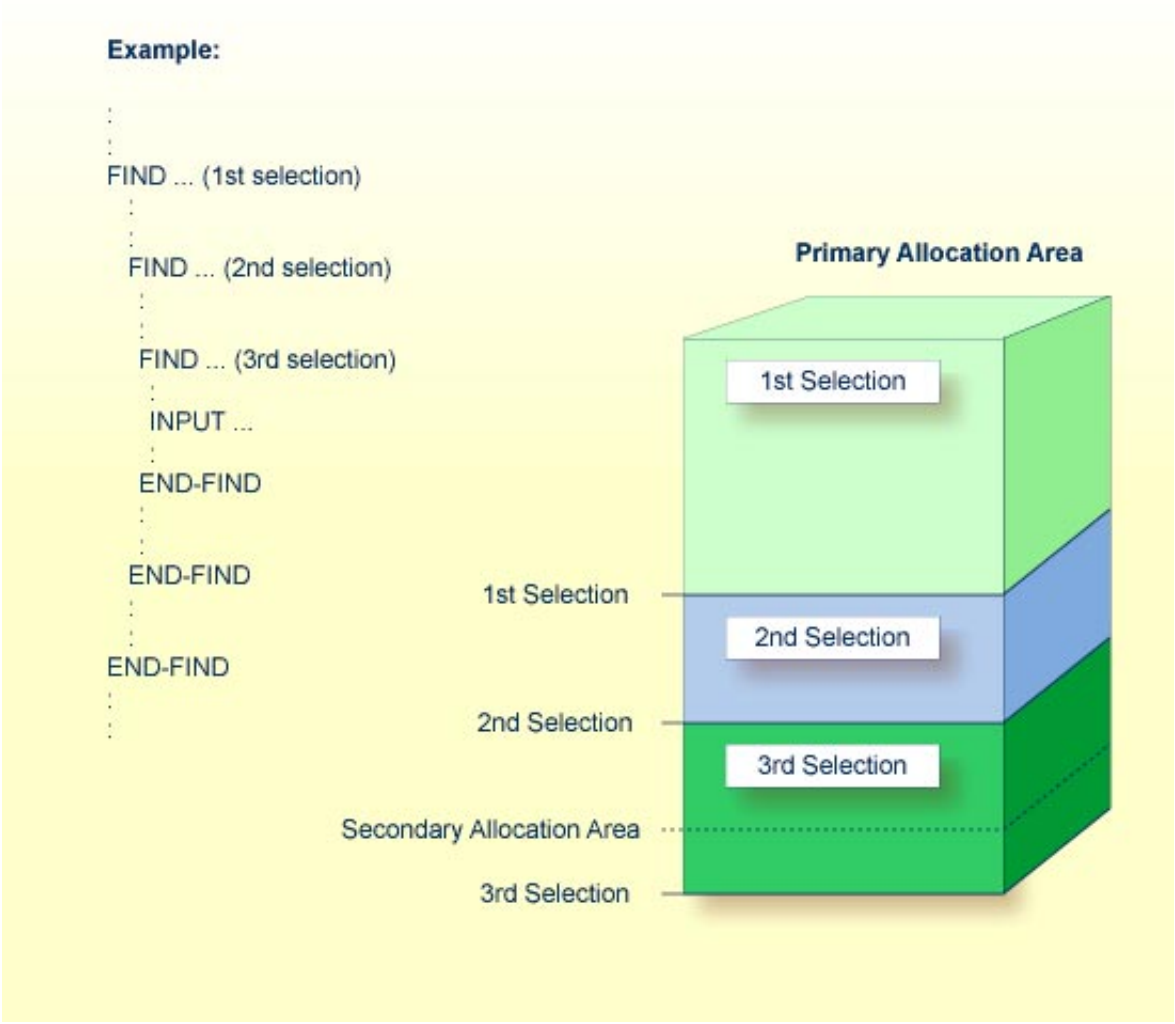
In the following example, the space allocated to the first selection is not released until all rows selected during the third selection have been retrieved. The same applies to the space allocated to the third selection.

The space allocated to the second selection, however, is released immediately after the last row of the corresponding selection result has been retrieved.

Therefore, the space allocated to the second selection can be used for the selection results of the third selection.



If the primary allocation area is not large enough, for example, if the third selection is nested within the second selection, the secondary allocation area is used.



When a session is terminated, all of a user's blocks are returned to the free blocks pool. If a session ends abnormally, Natural checks, where possible, whether a file server directory entry for the corresponding user exists. If so, all resources held by this user are released.

If Natural is unable to free the resources of an abnormally-ended user session, these resources are not released until the same user ID logs on from the same logical terminal again.

If the same user ID and/or logical terminal are not used again for Natural, the existing directory entry and the allocated space remain until the file server is formatted again. A new run of the formatting job deletes all existing data and recreates the directory.

NDB - Environment-Specific Considerations

Natural for DB2 can be run in the TP-monitor environments Com-plete, CICS, IMS/TM and TSO as well as in batch mode.

Under CICS, IMS/TM MPP and TSO, the Natural file server for DB2 is provided. The usage of the file server depends on the FSERV parameter in the NDBPARM parameter module as described in the relevant section.

- Natural for DB2 under Com-plete
- Natural for DB2 under IMS/TM
- Natural for DB2 under CICS
- Natural for DB2 under TSO
- Natural for DB2 using CAF
- Natural for DB2 using DB2 DL/I Batch Support

Natural for DB2 under Com-plete

DB2 is supported by Com-plete Version 4.5. Programs running under Com-plete 4.5 can access DB2 databases through the DB2 Call Attachment Facility (CAF). This facility, together with the Com-plete interface to DB2, allows fully conversational access to DB2 tables.

If the DB2 plan created during the installation process is not specified in your DB2 SERVER parameter list for Com-plete, you must explicitly call NATPLAN before the first SQL call to allocate this plan.

Natural for DB2 under IMS/TM

Under IMS/TM, Natural uses the IMS/DB2 Attachment Facility to access DB2. Therefore ensure that this attachment is started.

In IMS/TM transaction processing environments, DB2 closes all cursors and thereby loses all selection results whenever the program returns to the terminal to send a reply message. This operation mode is different from the way DB2 works in CICS conversational mode or TSO environments, where cursors can remain open across terminal communication and therefore selected rows can be retained for a longer time.

File Server under IMS/TM MPP

The file server is required to support the Natural for DB2 cursor management, while IMS/TM issues an implicit end-of-transaction to DB2 after each terminal I/O operation. With the file server, database loops can be continued across terminal I/Os, but database modifications made before a terminal I/O can no longer be backed out.

For a detailed description of the file server, refer to the section Natural File Server for DB2.

Natural for DB2 under CICS

Under CICS, Natural uses the CICS/DB2 Attachment Facility to access DB2. Therefore ensure that this attachment is started. If not, the Natural session is abnormally terminated with the CICS abend code AEY9, which leads to the Natural error message NAT0954 if the Natural profile parameter DU is set to OFF.

If your transaction ID is not assigned to any DB2 plan in the RCT, you must explicitly call NATPLAN before the first SQL call to specify the required DB2 plan. The actual plan allocation is performed by the dynamic plan selection exit.

Under CICS, a Natural program which accesses a DB2 table can also be run in pseudo-conversational mode. Then, at the end of a CICS task, all DB2 cursors are closed, and there is no way to reposition a DB2 cursor when the task is resumed.

To circumvent the problem of CICS terminating a pseudo-conversational transaction during loop processing and thus causing DB2 to close all cursors and lose all selection results, Natural either uses the file server to support the Natural transaction logic or switches from pseudo-conversational mode to conversational mode for the duration of a Natural loop which accesses a DB2 table.

If the file server is not used, Natural switches to conversational mode whenever a terminal I/O takes place during an open database loop.

To enable multiple Natural sessions to run concurrently, all Natural areas are written to the threads just before a terminal I/O operation is executed. When the terminal input is received, storage is acquired again, and all Natural areas are read from the threads.

File Server under CICS

In a CICS environment, the file server is an optional feature to relieve the problems of switching to conversational processing. Before a screen I/O, Natural detects if there are any open cursors and if so, saves the data contained by these cursors into the file server. With the file server, database loops can be continued across terminal I/Os, but database modifications made before a terminal I/O can no longer be backed out.

For a detailed description of the file server, refer to the section Natural File Server for DB2.

Natural for DB2 under TSO

Natural for DB2 can run under TSO without requiring any changes to the Natural/TSO interface.

Apart from OS/390 Batch, the batch environment for Natural can also be the TSO background, which invokes the TSO terminal monitor program by an EXEC PGM=IKJEFT01 statement in a JCL stream.

Both TSO online or batch programs can be executed either under the control of the DSN command or by using the Call Attachment Facility (CAF); the CAF interface is required if plan switching is to be used.

File Server under TSO

In a TSO environment, the file server is an optional feature to be able to emulate during development status a future CICS or IMS/TM production environment.

With each terminal I/O, Natural issues a COMMIT WORK command to simulate CICS or IMS/TM Syncpoints. Therefore, database modifications made before a terminal I/O can no longer be backed out.

For a detailed description of the file server, refer to the section Natural File Server for DB2.

Natural for DB2 using CAF

If you run Natural for DB2 under TSO or in batch mode and use the CAF interface, you must explicitly call NATPLAN before the first SQL call to allocate the required DB2 plan.

NATPLAN can be edited to specify the appropriate DB2 subsystem ID.

Natural for DB2 using DB2 DL/I Batch Support

If you want to access DB2 and DL/I in the same Natural session in batch mode (not BMP), you can use the DB2 DL/I batch support facility which allows you to coordinate recovery of both DB2 and DL/I database systems.

If you want to use this facility, you must execute the DLIBATCH procedure to run the DSNMTV01 module as the application program. DSNMTV01 in turn executes the Natural batch nucleus which must be linked with the DB2 interface DFSLI000.

Since DB2 does not allow Syncpoints when running in an DSNMTV01 environment, the Natural interface to DB2 cannot execute these Syncpoints by itself. These Syncpoints are therefore routed to the Natural interface to DL/I. Only one checkpoint is written for the corresponding END TRANSACTION statement, even if both database systems are involved.

If your PSB is generated with CMPAT=YES, all Syncpoints are executed by Natural for DL/I. Therefore, you must issue a END TRANSACTION statement before you end your Natural session; otherwise all your database modifications are lost, because Natural implicitly issues a BACKOUT TRANSACTION statement at the end of the session.

If your PSB is generated with CMPAT=NO, all Syncpoints are ignored.

